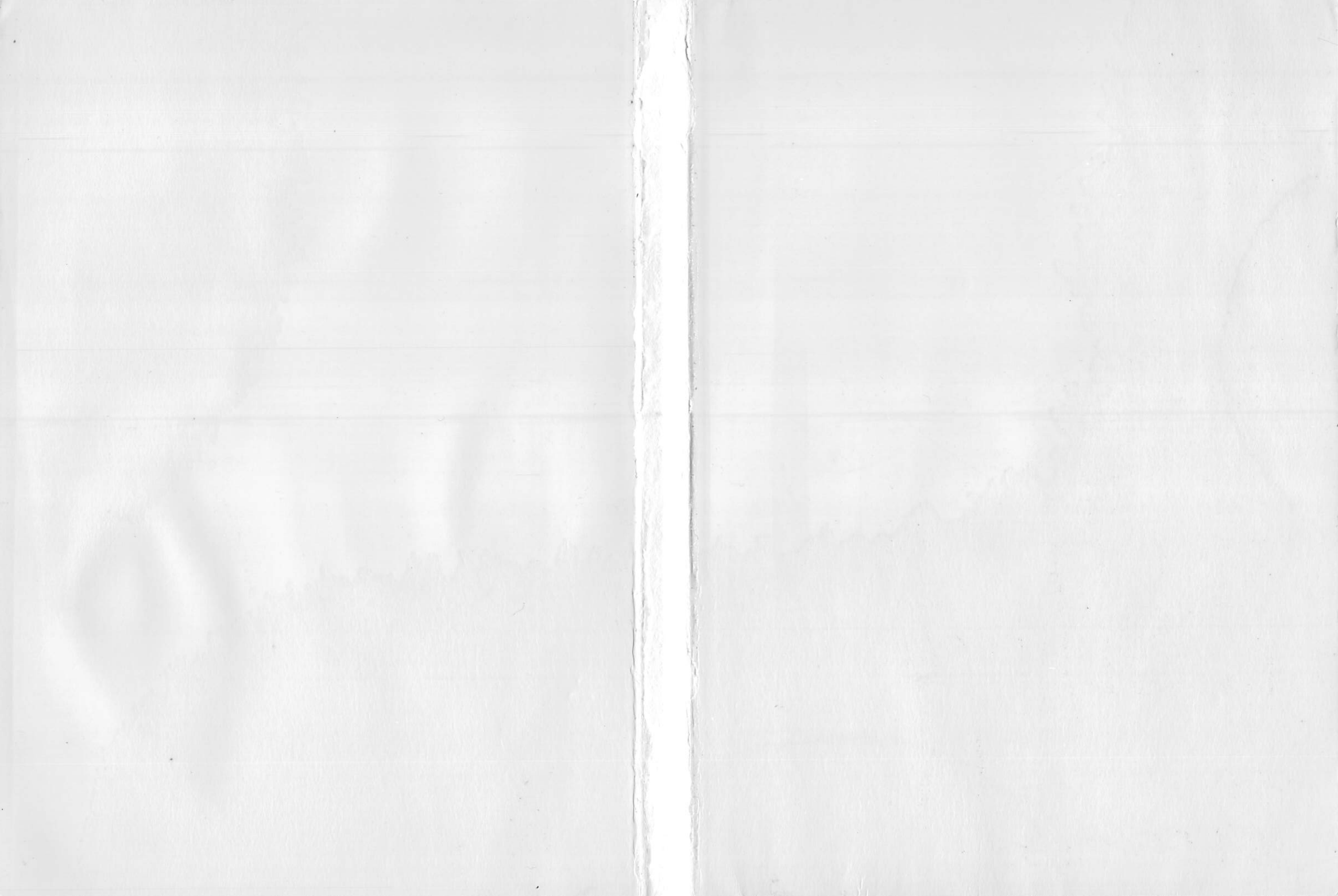
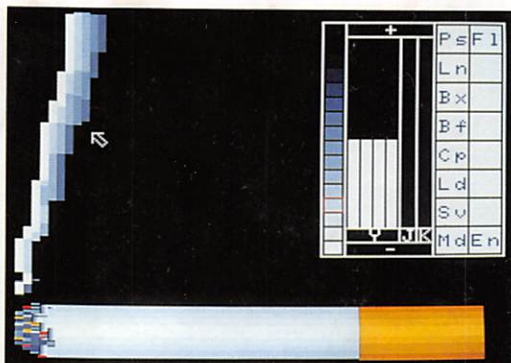
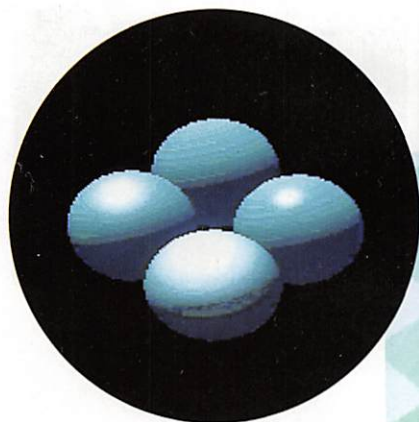
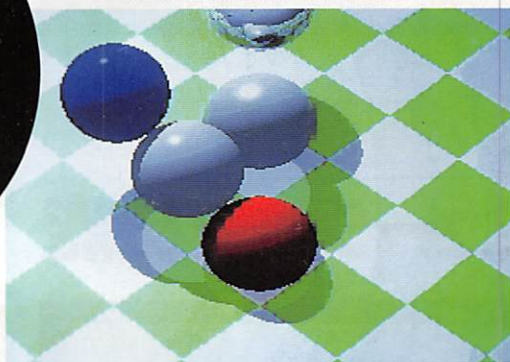
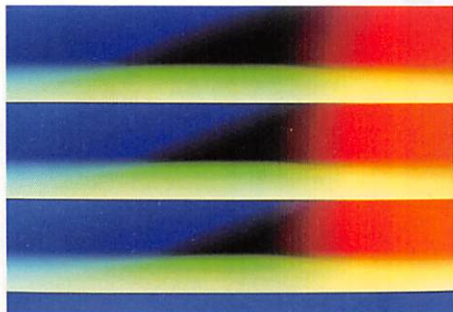
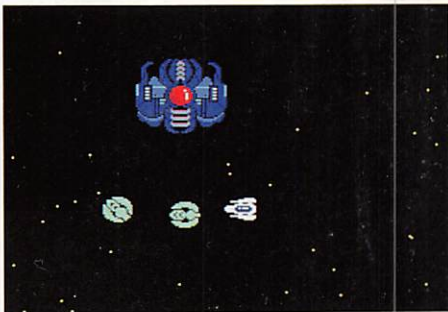
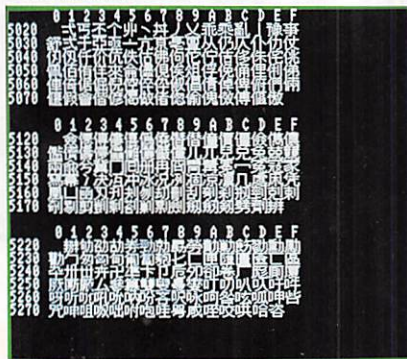
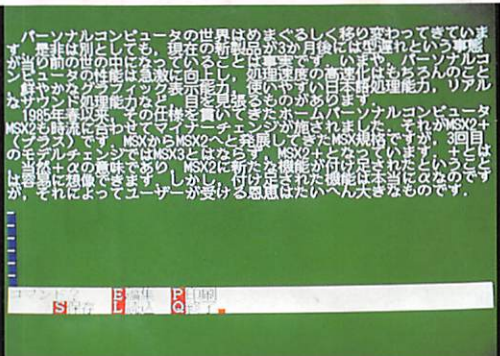
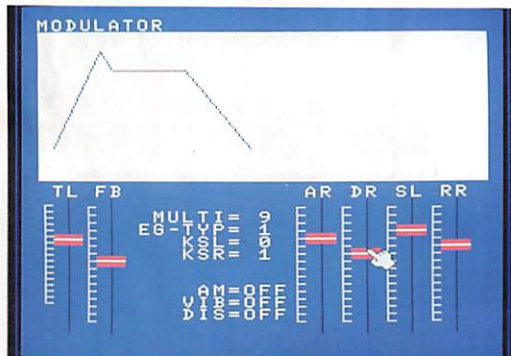
MSX 2+
パワフル活用法
杉谷成一著
MSX 2+ パワフル活用法
杉谷成一著
ASCII
ASCII BOOKS
アスキー出版局
ISBN4-87148-345-2 C3055 P1240E
定価1,240円(本体1,204円)

グラフィック機能

◀グラフィックエディタ(YJK モード)
YJKモードでは,横4ドット単位で描いていく.
YJKでの色の調合に若干のコツが必要だ／

▼グラフィックエディタ(RGB モード)
RGB モードでは,パレットカラーを1ドット単位で描くことができる.（本文 54 ページ）

▼レイトレーシング作品例
球と直方体のみでも，組み合せしだいでかなり楽しめる．気長に取り組む必要あり／？

▲レイトレーシングプログラム
光の反射を計算し，鮮やかな立体を描き出すぞ／（本文 43 ページ）

▼ 19,268 色のカラーグラーデーション
SCREEN12 の YJK モードを使った，カラー表示．19,268 色は目にも鮮やか／

▼ 8 方向スクロールのシューティングゲーム
ハードウェアスクロールによるゲーム．
撃って撃って，撃ちまくれ／（本文 183 ページ）

日本語処理機能

簡易漢字エディタ ▶
漢字が使えるテキストエディタ．表示文字数も自由に変えられる．（本文107ページ）

▼ JIS 第 2 水準漢字一覧表
MSX2＋は JIS 第 2 水準の難しい漢字もサポートしているのだ／ ROM さえあれば表示は簡単．

漢字スケジュール・メモ ▶
スケジュール管理はこれでバッチリ／約束の人名，地名も漢字で入力．（本文 98 ページ）

ミュージック機能

◀ FM 音色エディタ
リアルな FM 音源を使って，独自の音色を作り出せ／（本文 159 ページ）

FM 音源キーボード ▶
MSX2＋をミュージックキーボードに変身させて，華麗に演奏しよう／（本文 173 ページ）

MSX2+
パワフル活用法

杉谷成一著

アスキー出版局

商　標

MSX，MSX2，MSX2+ は株式会社アスキーの商標です．

〔Book Cover〕
Designed by STARKA
Illustrated by Nobuya Ito

本文イラスト　門山美知子

はじめに

MSX が統一仕様ホームパーソナルコンピュータとして発表されてから，もう5年の月日が経とうとしています．この間 MSX は，統一仕様である点や，手ごろな価格が評価され，色々な場面で活用されるようになってきています．

しかし，この間のパソコンの進歩はめまぐるしく，8ビットの時代から16ビットの時代へと，進んでいっています．

そんななかにあって，MSX は，パソコンとゲーム機の中間のマシンという位置付けになっているような気がします．この理由の1つには，MSX はいままで漢字 ROM がオプションだったことや，漢字表示の手順が標準化ができていなかったため，日本語対応のアプリケーション(いわゆる実用ソフトといわれるもの)があまり発売されていなかったことがあげられるでしょう．

今回発表された MSX2+では，漢字 ROM(第一水準)の標準化や，漢字処理機能が新設され，日本語処理にも十分対応できるようになりました．グラフィックも，いままで高級機種でなければ表示できなかった1万9千色同時表示になり，より自然画に近い表示ができるようになっています．

そして，ほとんどの MSX2+が，ディスクドライブ付きで，ワープロ機能を標準装備しており，なおかつ，6万円台という低価格で発売されています．しかし，低価格といっても，速度さえ気にしなければ，16ビットのパソコンにはできて，MSX にはできないことはほとんどありません．これは，MSX が基本的なパーソナルコンピュータの機能を備えているからこそできることだといえます．

今回の MSX2+へのステップアップによって，色数も漢字処理においても，他の8ビットパソコンに負けないものとなり，本当の意味でのホームコンピュータとなったといえるでしょう．

本書では，MSX2+で拡張された機能の解説や活用法を紹介していきます．みなさんの MSX2+活用の手助けになれば幸いです．

著者しるす

本書の構成

本書はMSX2＋の拡張された機能について解説するとともに，その機能を具体的に活用したプログラムを紹介しています．

本書は4部＋Appendixという構成になっており，2部以降はMSX2＋で拡張された機能ごとに分けられています．

さらに，各部はそれぞれ3つの章からなっています．1章目はそこで扱う機能に関する基礎的な知識，2章目はその機能を制御するために用意されたBASICコマンドのリファレンス，そして3章目はその機能を使う際の注意点と具体的な活用サンプルプログラムを紹介しています．

以下に，各部の具体的な内容を示します．

■ プロフィール

MSX2＋と従来のMSX2とではどこが違うのか？どんな機能が加わったのか？などの疑問に答えるのがこの部の目的です．ここでは，MSX2＋の特徴をかいつまんで紹介しています．

■ グラフィック機能

MSX2＋の最大の特徴であるグラフィック機能，とくに最大19,268色同時表示可能な新しいスクリーンモードを中心に紹介していきます．1，2章では，新しいスクリーンモードで使用されるグラフィックデータ方式のYJKと，これまでのRGB方式との違い，新しいVDPを搭載したことにより拡張されたMSX-BASIC Ver.3.0のコマンドなどを解説します．3章では，拡張されたVDPのレジスタについて触れたのち，活用サンプルとしてレイトレーシングとグラフィックエディタのプログラムを紹介しています．

■ 日本語処理機能

MSXがこれまで苦手としていた漢字の処理を可能にしたMSX漢字ドライバについて紹介していきます．1，2章では，MSXにおける日本語の表現の仕組みと漢字ドライバの役割，漢字ドライバ拡張BASICのコマンドなどを

解説します．3章では漢字ドライバ起動時の注意点をあげ，活用サンプルとして漢字スケジュールメモと漢字テキストエディタのプログラムを紹介しています．

■ ミュージック機能

MSX2＋と同時に発表されたFM音源の規格であるMSX-MUSICについて紹介していきます．1，2章では，FM音源そのものの仕組みと，MSX-MUSIC拡張BASICのコマンドについて解説してます．3章ではFM音源用ICであるOPLLの各レジスタについて触れたのち，活用サンプルとしてFM音色エディタとFM音源キーボードのプログラムを紹介しています．

■ Appendix

Appendix 1では，グラフィック機能で説明できなかったハードウェアスクロールについて，8方向スクロールのゲームを紹介しながら解説しています．

Appendix 2には，マシン語でプログラム組まれる方のために，MSX漢字ドライバとMSX-MUSICの拡張BIOSをリファレンス形式で掲載しています．

本書で使用したシステム機器構成

本体　　：MSX2＋（ディスクドライブ，MSX-MUSIC対応FM音源内蔵）
周辺機器：RGB対応カラーCRT

グラフィック機能の3章で紹介するレイトレーシングプログラムでは，「MSXべーしっくん」を使うと計算時間を短縮することができます．ただし，この場合には，計算精度の問題から画質が落ちることがあります．

ミュージック機能のところで紹介するプログラムは，すべてMSX-MUSIC対応のFM音源が必要です．内蔵されていない機種をお持ちの方は，別売のMSX-MUSIC対応のFM音源カートリッジをスロットに差してご利用ください．

Appendixの8方向スクロール・シューティングゲームはプログラム容量が大きいので，2ドライブのマシンでは正しく動作しないことがあります．そ

の場合は，リセットしてCTRLキーを押しながら起動してください.

なお，本書はMSX2＋の拡張された機能やコマンドにについてのみ触れております．したがって，従来のMSX2と互換のある部分につきましては，弊社既刊の「MSX2 BASIC入門」をお読みください.

目　次

MSX2+

プロフィール

MSX2+のパワフル機能

■ 誕生 MSX2+

パーソナルコンピュータの世界はめまぐるしく移り変わってきています．是非は別としても，現在の新製品が3か月後には型遅れという事態が当り前の世の中になっていることは事実です．いまや，パーソナルコンピュータの性能は急激に向上し，処理速度の高速化はもちろんのこと，鮮やかなグラフィック表示能力，使いやすい日本語処理能力，リアルなサウンド処理能力など，目を見張るものがあります．

1985年春以来，その仕様を貫いてきたホームパーソナルコンピュータMSX2も時流に合わせてマイナーチェンジが施されました．それがMSX2+(プラス)です．MSXからMSX2へと発展してきたMSX規格ですが，3回目のモデルチェンジではMSX3とはならず，MSX2+となっています．+とは，当然+αの意味であり，MSX2に新たな機能が付け足されたということは容易に想像できます．しかし，付け足された機能は本当にαなのですが，それによってユーザーが受ける恩恵はたいへん大きなものです．

■ MSX2+のシステム構成

MSX2からMSX2+へのバージョンアップの目玉は，なんといっても19,268色の色を同時表示できる，YJK方式の画面表示機能でしょう．YJK方式の詳細については次章で解説しますが，この機能は新しいVDP(ビデオディスプレイ・プロセッサ：画像表示処理LSI)によって実現されています．

さらに漢字処理機能が標準化され，誰にでも簡単に漢字を扱ったプログラムを使うことができるようになっています．

これは，MSX漢字ドライバというシステムソフトウェアによって実現されています．この漢字ドライバは，MSX2+では標準装備となっています．

漢字ドライバは，MSX-DOS2のカートリッジにも内蔵されていますし，

MSX-JE 規格の漢字変換フロントプロセッサと漢字ドライバを組み合わせたカートリッジも発売されていますので，これらのカートリッジをスロットに差すことによって，MSX2 でも漢字ドライバを使うことができます．このように，MSX2, MSX2＋ともに，統一された漢字入出力のための環境を作ることができます．

また，MSX2＋は，MSX, MSX2 との互換性を完全に保ったまま，バージョンアップされています．ですから，MSX, MSX2 用に作られたプログラムはすべて MSX2＋上で実行することができます．

それでは，MSX2＋で変更になった仕様について詳しく見ていくことにしましょう．

項目		MSX	MSX2	MSX2＋
CPU／速度		Z80／3.58MHz	⇦	⇦
メインメモリ容量(RAM)		8KB〜64KB	64KB〜4MB	⇦
システムプログラム容量(ROM)		32KB(MSX BASIC Ver1.0)	48KB(MSX BASIC Ver2.0)	96KB(MSX BASIC Ver3.0)
DOSプログラム容量(ROM)		16KB	⇦ 48KB(DOS2)	⇦ ⇦
画面表示機能	ビデオメモリ容量(VRAM)	16KB	128KB	⇦
	最大画面解像度(横×縦)	256×192	512×424	⇦
	最大同時表示色数	16色	256色	19,268色
	縦スクロール機能	なし	あり	⇦
	横スクロール機能	なし	なし	あり
オーディオ機能		PSG	PSG MSX-AUDIO(オプション)	⇦ ⇦ MSX-MUSIC(オプション)
日本語処理	漢字表示ROM	オプション	⇦	JIS第1水準標準
	漢字表示機能	アプリケーションによる	⇦	最大40字×24行
	漢字入力機能	アプリケーションによる	⇦	単漢変換標準・MSX-JE対応
通信機能	RS-232C	オプション	⇦	⇦
	モデム(300/1200BPS)	オプション	⇦	⇦

表 1.1　MSX2＋機能比較表

■ ハードウェア

MSX2ではVRAM(画像メモリ)は最低64Kバイトあればよかったのですが，近年のメモリ価格低下で，128Kバイトのメモリが本体価格に影響することはなくなりました．そこで，MSX2＋では画像メモリの大きさは，標準で128Kバイト固定になっています．メインメモリの量は，MSX2と変わっておらず，64Kバイト以上搭載されています．

また，JIS第一水準漢字ROMが標準装備になりました．

その他の周辺機器とのインターフェイスに関してはMSX2と同じです．

■ 新しいグラフィック機能

MSX2＋では，新しいVDPの採用により，これまでのMSX2に比べて次のような機能が追加されています．

今までと同じ容量の画像メモリで，表現できる色数を増やした，YJK方式による自然画表示モードが追加されています．このYJK方式の利用で，19,268色または12,499色を同時発色することができます．

MSX2では垂直方向のハードウェアスクロールが可能でしたが，MSX2＋では垂直方向に加えて，1ドット単位で水平方向にハードウェアスクロールができるようになっています．したがって，これからは横スクロールタイプのゲームを簡単に作ることができます．

また，今までは，ビットマップモード(グラフィックモード)でのみ実行可能だった，VDPコマンドが，すべてのスクリーンモードにおいて使えるようになりました．

その他，インターフェイスに関して細かい機能の拡張がなされています．

■ MSX-BASIC Ver.3.0

MSX-BASICは，MSX-BASIC Ver.2.0の上位互換性を保ちつつ，グラフィック処理に関する拡張がなされ，Ver.3.0となっています．

VDPの変更に伴い，BASICからVDPの新機能であるYJK方式の画面表示やドット単位の横スクロールを制御できるようにしています．

YJKモード用のグラフィック画面として，SCREEN10～12までが新設されています．それに伴い，YJKモードに対するグラフィックコマンドや，

BASE関数，VDP関数の拡張が行われています．

縦スクロールと横スクロールをBASICから制御するコマンドも新設されています．

■ MSX漢字ドライバ

MSX漢字ドライバをシステムに組み込むことによって，BIOSを使用して文字を入出力しているプログラムならば，変更なしで漢字入出力が可能になります．したがって，BASICはもちろん，MSX-DOS上でも漢字の入出力が可能です．また，JIS第1，第2水準の漢字をサポートしています．

システムに仮想端末入力インターフェイス(MSX-JE)が存在すれば，その漢字変換プロセッサを使って漢字の入力を行うことができます．もしMSX-JEが存在しない場合でも，漢字ドライバに内蔵されている単漢字変換機能によって，漢字の入力が可能です．

漢字プリンタへの出力もサポートしていますから，漢字のメッセージを含んだプログラムのリストを打ち出したり，漢字のメッセージをプリンタにプリントすることができます．グラフィック画面に対しても，LOCATE文，PRINT文等で漢字のメッセージを出力できます．

さらにBASIC上で全角半角の混ざった文字列を正しく処理するためのコマンドを拡張BASICの形で用意しています．

■ MSX-MUSIC(オプション)

FM音源を使ったサウンド機能です．メロディ音9音，あるいは，メロディ音6音＋リズム5音を発音することができます．MSX-MUSICを使えば，簡単にゲームの効果音やBGMを作ることができます．

MSX-MUSIC拡張BASICでは，PLAY文とMML(ミュージックマクロランゲージ)で，FM音源9音＋PSG音源3音，あるいは，FM音源6音＋リズム音＋PSG音源3音を同時に鳴らすことができます．

この規格は基本的にはオプションです．もし内蔵されていなくても，松下電器から発売されている，「パナ・アミューズメント・カートリッジ(FM-PAC)」を使えばMSX-MUSICとして使うことができます．

MSX2+

グラフィック機能

MSX2+で拡張されたグラフィック機能

MSX2からMSX2+への最も大きな変更点はVDPです．ここでは，新しいVDP，V9958により飛躍的に向上したグラフィック処理機能について解説していきます．

■ スクリーンモードの追加

MSX2+では，新たにSCREEN10からSCREEN12までの3つのスクリーンモードが追加されています．これは新しいVDPの機能を使った自然画表示モードで，とくにSCREEN12では最大19,268色同時表示が可能となっています．

■ 縦横ハードウェアスクロール

MSX2では，ハードウェアでサポートされる画面のスクロールは縦方向のみでした．したがって，画面を横方向にスクロールさせるためには，すべてプログラムで行わなければなりません．しかし，MSX2+では，縦方向に加えて横方向のスクロールもハードウェア･レベルでサポートしているため，BASICのコマンド1つで，ドット単位の横スクロールも高速に行うことができます．

この機能により，いままでは横スクロールのゲームといえばSCREEN2を使ったものでしたが，これからは，SCREEN5以上のモードでも簡単に横スクロールさせることができることになります．

従来の画像データ処理方式

MSX2のVRAMを128Kバイト搭載したモデルでは，最大256色同時表示が可能でした．ところが，MSX2+ではVRAMを同じ128Kバイトしか搭

載していないにもかかわらず，19,268色という膨大な色数を同時に表示することができます．これは，新しい画面モードでの画像データのVRAMへの格納方式が，従来のMSX2の方式とはまったく異なっており，メモリを効率よく使用しているためです．

ここで，MSX2とMSX2＋の画像データの処理の違いについて見ていくことにしましょう．

■ RGB方式

普通，コンピュータで色の情報を扱うには，表示する画像をRed(赤)，Green(緑)，Blue(青)に分解して，その割合を数値として扱います．このR, G, Bの値を画素ごとに，そのままVRAM上に格納してしまうのが最も一般的な画像データの格納方式です．SCREEN8もこの方法を採用しています．

この方式では，R，G，Bそれぞれの割合を細かくしていけばたくさんの色を表現していくことができるわけですが，そのためには当然大容量のメモリが必要になります．

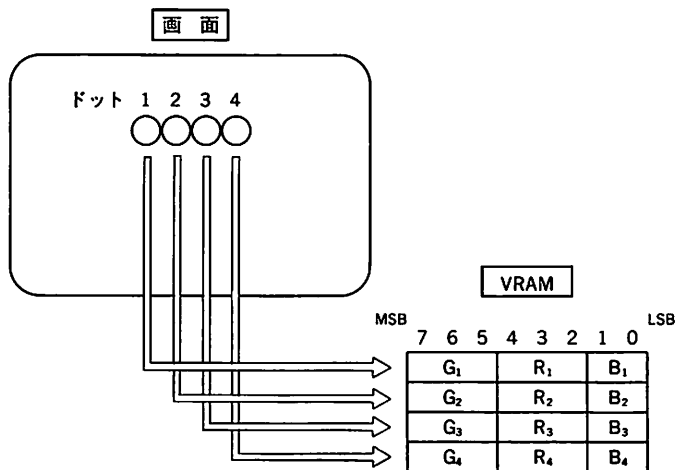



図2.1 画素ごとのRGBデータの格納

VRAMを効率よく使う方法としては，パレットを使う方法が一般的で，これはMSXのSCREEN7でも使っています．絵を描くときのパレットと同じように，色のデータを蓄えておくところを作り，そこに番号を振り，このパレットにRGBの3つの値を与えて色を決めます．VRAMにはパレットの番号を書いておいて，表示するときにVRAMに書かれた番号に対応するパレットに設定された色を出力します．この方法だと，パレット側の分解能を上げるだけで表示できる色数が増えるわけです．しかし，一度に表示できる色はパレットの数だけなので，パレットの数を増やせば必然的にVRAM容量を大きくしなければならなくなります．

しかし，大容量のメモリを搭載すれば，そこにデータを書き込んでいるCPUの負担が増えて処理が遅くなってしまったり，メモリ分のコストがかかってしまうため，MSXではあまりよい方法とはいえません．

そこで，VRAMの容量を増やさずに，表示できる色数を増やすために，MSX2+では，YJK方式というデータ形式を採用しています．これは，従来の方法とは異なり，RGBのデータを圧縮し，データ量を少なくしてVRAM

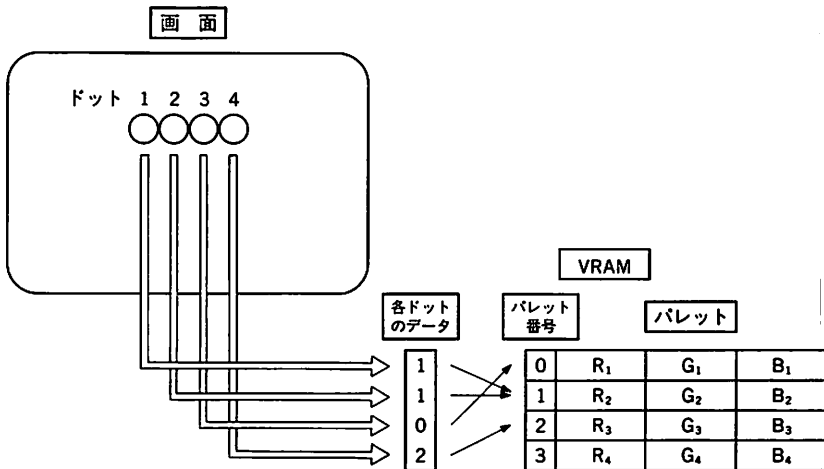



図2.2　パレットを使ったカラーグラフィック表示

に蓄えておき，表示するときに圧縮されたデータをもとのRGBに戻して表示する方法です．

このYJK方式は，テレビ放送などにも使われているNTSCビデオ方式と深い関係があります．そこで，YJK方式の説明にはいるまえに，NTSCについて簡単に触れておきましょう．

■ ビデオ信号の圧縮

現在のテレビ放送は白黒テレビでも受像することができます．これはテレビが白黒からカラーへ発展する段階で，送信する信号に互換性を持たせたためです．

白黒テレビの画像は，光の明るさだけが変化します．ですから，白黒テレビしかなかった時代のテレビ放送は，光の明るさ(輝度といいます)の信号を映像信号として送っていました．

ところが，カラーテレビカメラでは，レンズからはいってきた光を，フィルターを使ってRGBの3原色に分割し，それぞれの光を3つの撮影管で撮影して3種類の映像信号(RGB)を作り出します．受像機側も，RGBそれぞれのデータを受け取って，それらを合成してカラー画像を再現します．したがって，本来ならばRGBのデータをそのまま映像信号として送ればよいのですが，そうすると輝度情報が3つのデータに分けられてしまうため，白黒テレビでは受像できなくなってしまいます．

そこで，白黒テレビでもカラーテレビでも受像できるように考え出されたのが，NTSC方式と呼ばれるデータ形式です．

■ NTSC方式

RGBの3つの信号からRの30%，Gの59%，Bの11%を取り出して合成すると，ちょうど白黒テレビカメラと同じ感度特性の信号が得られます．これを輝度信号(Yという)として使います．

カラーテレビの場合，さらに色に関する情報が必要です．そこで，輝度(Y)情報に加え，色相や彩度(飽和度)といった色の情報を，各RGB信号から輝度(Y)を引くことにより求めます(これを色差信号と呼びます)．

$$Y = 0.30R + 0.59G + 0.11B$$
$$R - Y = 0.70R - 0.59G - 0.11B$$
$$B - Y = -0.30R - 0.59G + 0.89B$$
$$G - Y = -0.30R + 0.41G - 0.11B$$

このなかでG-Yに関しては，その他の式から導き出せるので，最終的にはG-Yを除いた3つを使って色を表現します．R-Y, B-Yのことを色差信号といいます．

人間の目は，色に対しての分解能はあまりよくありません．ある程度細かくなると，その色を識別することができなくなってしまいます．したがって，ある面積以下に色を付けても意味がありません．そこで，NTSCでは比較的面積の大きいところにだけ色が付くように色差信号を操作し，情報量を少なくしています．

こうしてできた3つの信号を，お互いに影響しないように加え合わせて，1つの信号として送信します．これらの信号を受け取ることにより，白黒テレビは輝度信号で，カラーテレビは輝度信号＋色差信号でそれぞれ画像を再現することができます．

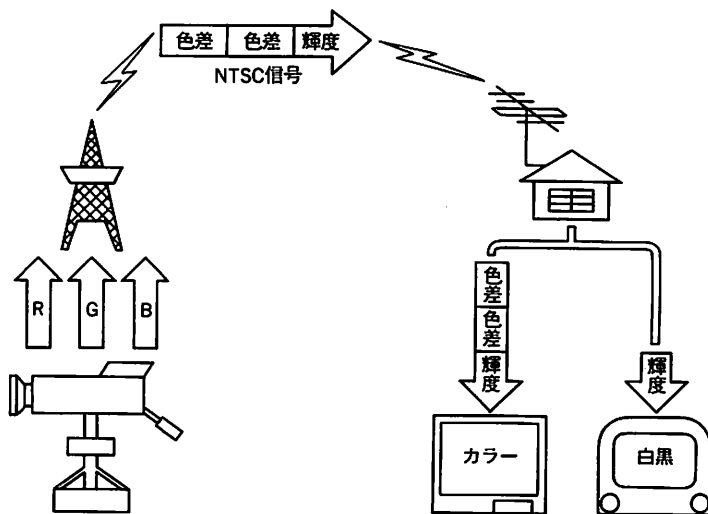



図 2.3　NTSC方式での送信

YJK方式による画像データ処理

MSX2＋の画像データ処理法であるYJK方式とは，NTSC方式をまねて，色を輝度信号と色差信号に分け，さらに色の付く面積を制限して情報量を少なくして，輝度情報は1ピクセルごとに，色情報である色相，彩度(飽和度)の情報は4ピクセルごとに設定できるようにしたものです．

■ YJK方式でのVRAM上のデータ形式

YJK方式では，RGBを輝度と色相，彩度(飽和度)に分けて，そのデータをVRAM上に置き，表示するときにRGBに変換して出力する方法をとっています．また，情報量を少なくするために，輝度は1ドットごとに設定できるのですが，色情報に関しては4ピクセルごとにしか設定できないようになっています．

RGBからYJKへの変換は下の式のようになっています．

$$Y = \frac{1}{2}B + \frac{1}{4}R + \frac{1}{8}G$$
$$J = R - Y$$
$$K = G - Y$$

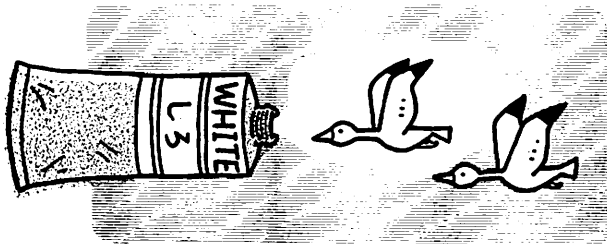
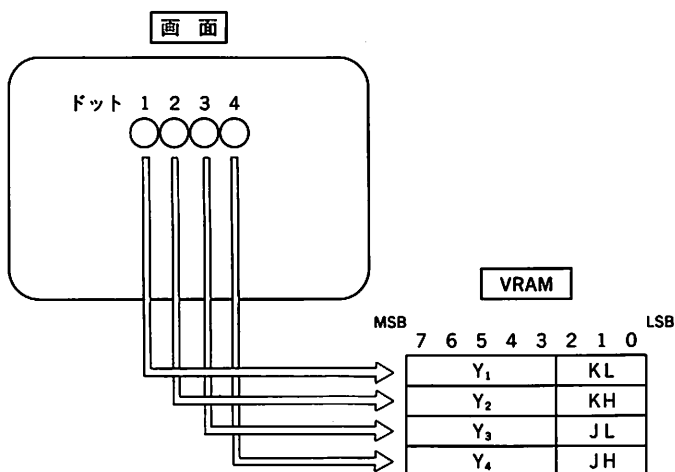
この式から，Yは輝度情報，JKは色差情報になっていることがわかります．Yを生成する式の係数は，デジタルでの計算のしやすさと，実際の表示での発色のよしあしを考慮して決められたものです．

実際には，SCREEN8と同じ解像度のモード(256×212ドットで1ドットあたり8ビットの情報を持つモード)で，横4ドットを1組として色を表現します．SCREEN12のYJKモードのとき，実際のVRAM上のデータの内容は下図のようになっています．

・Ynは横4ドット中n番目のドットの輝度
KLはKの下位3ビット，KHはKの上位3ビット
JL，JHについても同様
Yのとる値の範囲は0から31まで，JKのとる値は符号付きで，
それぞれ－32から31まで

図2.4 YJKモード時のVRAM上のデータ

V9958では，YJKそれぞれの値をVDP内部で下のような操作をしてRGB出力しています．

$$R = Y + J$$
$$G = Y + K$$
$$B = \frac{5}{4}Y - \frac{1}{2}J - \frac{1}{4}K$$

画面上に表示される4ドットのRGBデータは次のように計算されます．

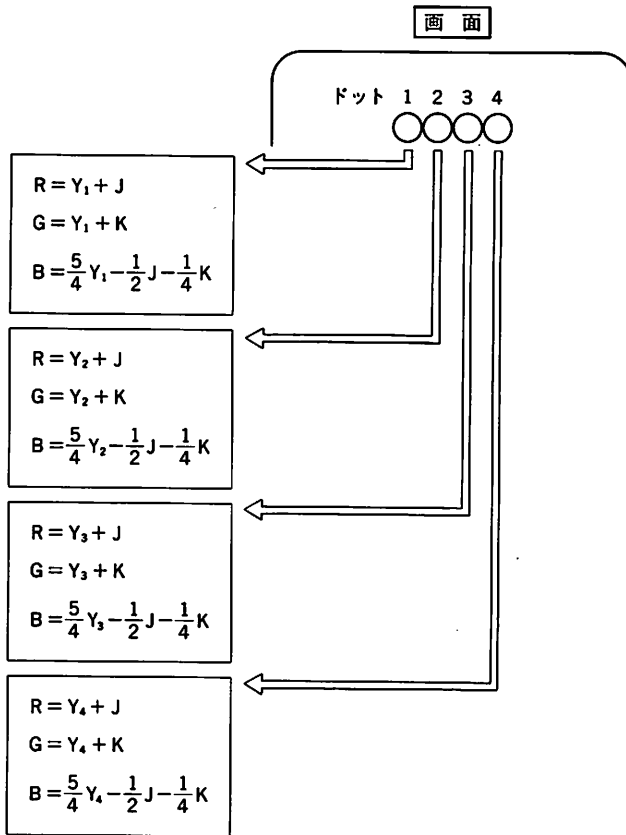



図2.5　横4ドットのデータ

このことから，Y(輝度)は，1ドットごとに独立して設定することができることがわかると思います．

YJKモードで出力できる色の数は，19,268色となっています．しかし，この数はYJKの値の範囲から考えると異常に小さい数字です．理論的には，131,072色発色できるはずです．これは，VDPが計算結果を5ビットで処理していることと，VDPがデジタルで計算しているので，余りを切り捨ててしまうことから起きるようです．

この方式の最大の長所は，今までと同じ情報量で，より多くの色を扱えるということでしょう．

しかし，色の情報が4ドットごとにしか付けられないことから，なめらかに色が変化するものはきれいに表現できますが，色の変化が激しいもの，たとえば1ドットごとに色が大きく変わるものや幾何学的な模様などをきれいに表示することはできません．このため自然画などに向いた，自然画モードなどと呼んでいます．しかし，実際には，気になるほど色の境目が目だつことはないようです．

■ アトリビュート混在のYJKモード

上記のYJK方式では，4ドットごとに色が付くために，グラフィックスを表示する場合はよいのですが，文字などを一緒に表示するのは難しくなります．

そこで，Yデータの長さを1ビット減らして，その1ビットをアトリビュートビットとして設定し，そのビットの状態により，ドットごとにRGBで色指定ができるモードを用意しています．

具体的には，アトリビュートビットが1の場合は，Yの4ビットをパレットコードとして扱い，パレットに設定したRGBのデータに基づき，色を表示しています．アトリビュートビットが0の場合は，YをそのままYJKのデータとして色を表示します．

こうすることによって，1ドット単位でYJKモードとRGBモードを共存させることができるようになります．したがって，YJKの4ドットの境界を気にすることなく漢字やキャラクターを表示できます．また，LINEやCIRCLE命令も実行することができます．

このモードをアトリビュート混在のYJKモードと呼び，SCREEN10と11がこれに当たります．また，これに対してSCREEN12のときのモードをアトリビュートなしYJKモードあるいはたんにYJKモードと呼んでいます．

アトリビュート混在モードでのVRAM上のデータの形式を図2.6に示します．

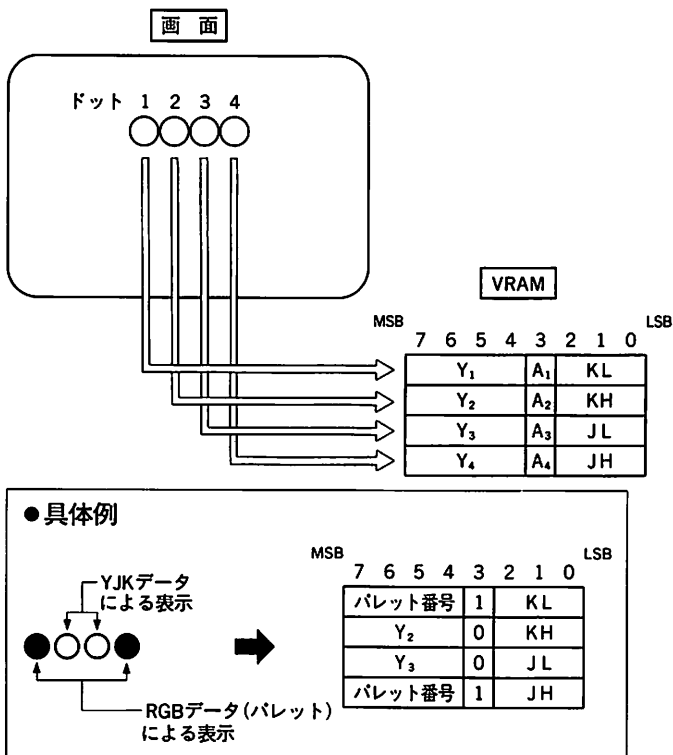
アトリビュート混在YJK方式からRGBへの変換も基本的には同じで，Yは輝度，JKは色差情報にあたります．

・Anは横4ドット中n番目のドットのアトリビュート
Ynは Anが0のときn番目のドットのYJKデータを表わし，Anが1のときパレット番号を表わす．

図 2.6 アトリビュート混在の YJK モード時の VRAM 上のデータ

2 MSX-BASIC Version3.0 VDP関連 コマンドリファレンス

MSX-BASIC Version 3.0 で，Version 2.0 から変更のあったコマンドと，新設されたコマンドについて，コマンドごとに解説していきます．変更されたコマンドも，そのほとんどが新 VDP の機能に関するものです．また，新設されたコマンドは，スクロールに関するもの1つだけです．

以下に変更・新設されたコマンドをあげておきます．

SCREEN
BASE
SET SCROLL
PUT KANJI
VDP

SCREEN

●書式● SCREEN [<画面モード>] [,<スプライトサイズ>]
[,<キークリックスイッチ>] [,<カセットボーレート>]
[,<プリンタオプション>] [,<インターレースモード>]

MSX2 の SCREEN1～SCREEN8 に加えて，YJK モード用の 10～12 までが使えます．SCREEN9 は海外仕様との関係上，欠番になっています．<画面モード> 以外のスイッチは BASIC Ver.2.0 と同じです(表 2.1)．

新設された各モードの意味を解説します.

モード	スクリーン形式	ドットまたは文字数	同時表示色
0	テキスト	80(40)×24文字	512色中2色
1	テキスト	32×24文字	512色中16色
2	高解像度グラフィック	256×192ドット	512色中16色
3	低解像度グラフィック	64×48ドット	512色中16色
4	高解像度グラフィック	256×192ドット	512色中16色
5	ビットマップグラフィック	256×212ドット	512色中16色
6	ビットマップグラフィック	512×212ドット	512色中4色
7	ビットマップグラフィック	512×212ドット	512色中16色
8	ビットマップグラフィック	256×212ドット	256色
10	アトリビュート混在YJK	256×212ドット	12,499色
11	アトリビュート混在YJK	256×212ドット	12,499色
12	アトリビュートなしYJKモード	256×212ドット	19,268色

*SCREEN 9は使われていません.

表 2.1 MSX2+の画面モード一覧

SCREEN 10(アトリビュート混在の YJK モード)

アトリビュート混在の YJK モードを選択します. このモードで指定できる色コードの範囲は 0～15 までです.

LINE 文や PAINT 文等のグラフィック描画の際の処理は, どんなロジカルオペレーションを指定したかにより動作が違います. 以下のように変わります.

①ロジカルオペレーションが PSET または省略された場合

PSET や LINE 文を使って描画する場合には, 対象となる画素(1バイト)のアトリビュートビットがセットされ, 上位の 4 ビットが指定された色コードとなります. この場合, 指定した色コードはパレットコードとして解釈されます.

②ロジカルオペレーションに PSET 以外が指定された場合

指定された色コードを左に 4 ビットシフトした値と, 対象となる画素との

あいだで指定されたロジカルオペレーションが実行されます．この場合，指定した色コードは，アトリビュートビットがセットされている画素に対してはパレットコードとして，アトリビュートビットがセットされていない画素に対しては，Y成分として指定していることになります．

SCREEN11（アトリビュート混在のYJKモード）
アトリビュート混在のYJKモードを選択します．このモードで指定できる色コードの範囲は0から255までです．SCREEN10と違い，描画の際に指定された色コードはそのまま指定された画素のデータとして演算されます．

SCREEN12（完全YJKモード）
アトリビュートなしのYJKモードを選択します．このモードで指定できる色コードの範囲は0から255までです．描画の際に指定された色コードは，そのまま指定された画素のデータとして演算されます．

従来のSCREEN文では，スクリーンモードが変わるたびに表示を初期化しましたが，SCREEN10, 11, 12の間での遷移では表2.2のようになります．

表示が初期化されない遷移
SCREEN10 から SCREEN11
SCREEN10 から SCREEN12
SCREEN11 から SCREEN10
SCREEN11 から SCREEN12

表示が初期化されずアトリビュートビットが0になる遷移
SCREEN12 から SCREEN10
SCREEN12 から SCREEN11

表2.2　SCREEN10〜12でのスクリーンモードの変更に伴う状態遷移

リスト 2.1　SCREEEN12 でカラー・グラデションーンを表示

```
100 SCREEN 12:CLS
110 FOR Y=0 TO 31 STEP 3
120   YD=0:CLS
130   FOR I=0 TO 2
140     FOR K=-32 TO 31
150       XD=0
160       FOR J=-32 TO 31
170       PSET(XD+0,YD),(Y+I)*8+(K AND 7)
180       PSET(XD+1,YD),(Y+I)*8+(K AND 56)/8
190       PSET(XD+2,YD),(Y+I)*8+(J AND 7)
200       PSET(XD+3,YD),(Y+I)*8+(J AND 56)/8
210       XD=XD+4
220       NEXT J
230     YD=YD+1
240     NEXT K
250   NEXT I
260   IF INKEY$="" THEN 260
270 NEXT Y
280 END
```

リスト 2.2　SCREEN10 の LINE 命令でのロジカルオペレーションの実行

```
100 SCREEN 12
110 J=-32
120   FOR YA=0 TO 191 STEP 64
130     FOR XA=0 TO 7
140       YD=YA
150       FOR K=-32 TO 31
160         XD=XA*32
170         FOR Y=0 TO 31 STEP 4
180           PSET(XD+0,YD),(Y+0)*8+(K AND 7)
190           PSET(XD+1,YD),(Y+1)*8+(K AND 56)/8
200           PSET(XD+2,YD),(Y+2)*8+(J AND 7)
210           PSET(XD+3,YD),(Y+3)*8+(J AND 56)/8
220           XD=XD+4
230         NEXT Y
240         YD=YD+1
250       NEXT K
260       J=J+1
```

```
270      NEXT XA
280    NEXT YA
290 SCREEN 10
300 C=0
310 FOR Y=0 TO 211 STEP 32
320   FOR X=0 TO 255 STEP 32
330      LINE (X,Y)-(X+15,Y+15),C,BF,PSET
340      LINE (X+16,Y+16)-(X+31,Y+31),C,BF,PSET
350      C=C+1:IF C=16 THEN C=0
360   NEXT X
370 NEXT Y
380 C=15
390 FOR Y=0 TO 211 STEP 64
400   FOR X=0 TO 255 STEP 64
410      LINE (X,Y)-(X+31,Y+31),C,BF,OR
420      LINE (X+32,Y+32)-(X+63,Y+63),C,BF,OR
430      C=C-1:IF C=-1 THEN C=15
440   NEXT X
450 NEXT Y
460 IF INKEY$="" THEN 460
```

BASE

●書式● BASE(<数式>)

画面出力に関連するテーブル(VRAM上のテーブル)の先頭アドレスを求めます.

<数式>で指定されたVRAM上のアドレスを出します. 指定できる数値は, 0から44および50から64までです. <数式>と各画面モードのテーブルは表2.3のように対応しています. 45から49を指定することはできません.

たとえば, SCREEN8のスプライトアトリビュート・テーブルの先頭アドレスを求めるには次のように指定します.

```
PRINT  BASE(43)
```

表2.3に, BASEコマンドで指定する数式と各画面モードのテーブルの対応を示しておきます.

数式	画面モード	テーブル
0	0	パターンネーム・テーブル
1	0	カラーテーブル
2	0	パターンジェネレータ・テーブル
3	0	スプライトアトリビュート・テーブル
4	0	スプライトジェネレータ・テーブル
5	1	パターンネーム・テーブル
6	1	カラーテーブル
7	1	パターンジェネレータ・テーブル
8	1	スプライトアトリビュート・テーブル
9	1	スプライト・ジェネレータ・テーブル
10	2	パターンネーム・テーブル
11	2	カラーテーブル
12	2	パターンジェネレータ・テーブル
⋮	⋮	⋮
43	8	スプライトアトリビュート・テーブル
44	8	スプライトジェネレータ・テーブル
45～49		空き
50	10	パターンネーム・テーブル
51	10	カラーテーブル
52	10	パターンジェネレータ・テーブル
53	10	スプライトアトリビュート・テーブル
54	10	スプライトジェネレータ・テーブル
55	11	パターンネーム・テーブル
56	11	カラーテーブル
57	11	パターンジェネレータ・テーブル
⋮	⋮	⋮
63	12	スプライトアトリビュート・テーブル
64	12	スプライトジェネレータ・テーブル

表 2.3　BASE コマンドでの指定と各画面モードのテーブル

SET SCROLL

●書式● SET SCROLL [〈X〉] [, 〈Y〉] [, 〈MASK〉]
[, 〈2PAGE〉]

表示画面を水平方向および垂直方向にスクロールします．〈X〉，〈Y〉に数式を指定することにより，それぞれ水平方向，垂直方向のスクロール量を指定します．〈X〉のとる値の範囲は0から511，〈Y〉のとる値の範囲は0から255までです．

SCREEN1～4，SCREEN8～12では水平方向に1ドット単位で，またSCREEN6，7では水平方向に2ドット単位でスクロールします．

〈MASK〉は表示画面の左端8ドット(SCREEN6，7では16ドット)をボーダー色で隠すかどうかを数式で指定します．〈MASK〉が0のときは隠さず，1のときは隠します.8の倍数ドット単位でスクロールさせる場合以外はマスクしたほうが美しく見えます．

〈2PAGE〉は水平スクロールをしたときに，表示内容の左端および右端にどのページを表示するかを数式で指定します．

〈2PAGE〉が0ならば，水平スクロールをしたときにそのページ内で表示がラップします．つまり，画面の左端に消えていった画面がそのまま右端から表示されてきます．

〈2PAGE〉が1ならば，2ページ連続の水平スクロールになります．つまり,ページ0の画面が画面の左端に消えていくと同時に,右端からページ1の画面が表示されてきます．また，このときは表示ページをSET PAGEステートメントにより奇数ページにする必要があります．

SCREEN0の画面では，SET SCROLLの動作が異なります．水平スクロールは〈X〉の下位3ビットのみが有効となります．〈X〉が1のときは7ドット，2のときは6ドット，3のときは5ドット，……，7のときは1ドットそれぞれ左にスクロールします．7ドット以上はスクロールしません．

垂直スクロールも同様で〈Y〉の下位3ビットのみが有効となります．〈Y〉が1のときは1ドット，2のときは2ドット，3のときは3ドット，……，7のときは7ドットそれぞれ上にスクロールします．7ドット以上はスクロー

ルしません．また，このとき表示されている文字の位置自体は変更されておらず，文字のフォントの表示開始ラスタが変わっていきます．

スクロールさせるというのは，VRAM内の走査開始アドレスをずらすだけです．つまり画面に表示される位置が変わるだけで，XY座標も一緒に移動していきますので，プログラムを組む際には注意が必要です．

また，スクロールさせることによって，スプライトの表示位置も調整が必要になります．Y方向へのスクロールでは，スプライトの位置も移動してしまいます．X方向へのスクロールでは，スプライトのディスプレイ上での位置は変わりません．

SCREEN8やSCREEN7の場合，256×256または512×256フルに垂直スクロールさせるとすると，64Kバイトフルに画像データ領域として使用しなければなりません．SCREEN8は，1バイト1ドットで256×256×1バイト＝64Kバイト，SCREEN7では，512×256×0.5バイト＝64Kバイトとなって，1ページ分のVRAMをすべて使わなければならないため，スプライトアトリビュートテーブルなどのスプライト用テーブルを取る場所がなくなってしまいます．スプライトを使用する場合は注意が必要です．

リスト2.3　水平／垂直スクロールとスプライトの表示

```
100 SCREEN 5,2:SET PAGE 0,0:CLS
110 FOR YD=0 TO 255
120   FOR XD=0 TO 127
130     C=INT(RND(1)*16):VPOKE YD*128+XD,C*16+C
140   NEXT XD
150 NEXT YD
160 SET PAGE 1,1:SCREEN ,2:CLS:SPRITE$(0)=STRING$(32,CHR$(&HFF))
170 I=0:FOR Y=1 TO 4:FOR X=1 TO 4
180 PUT SPRITE I,(32*X,32*Y),8,0:I=I+1
190 NEXT X:NEXT Y
200 A=VDP(12):B=VDP(5):C=VDP(6)
210 SET PAGE 0,0
220 VDP(12)=A:VDP(5)=B:VDP(6)=C
230 T=STICK(0)
240 X=X-(T>1 AND T<5)+(T>5 AND T<9)
250 IF X>511 THEN X=0
260 IF X<0 THEN X=511
270 Y=Y-(T>3 AND T<7)+((T=1 OR T=2)OR T=8)
```

```
280 IF Y>255 THEN Y=0
290 IF Y<0 THEN Y=255
300 SET SCROLL X,Y,1
310 GOTO 230
```

リスト2.4 SCREEN12での2画面スクロール

```
100 SCREEN 12
110 J=-32
120 FOR I=0 TO 1
130   SET PAGE I,I:CLS
140   FOR YA=0 TO 255 STEP 64
150     FOR XA=0 TO 7
160       YD=YA
170       FOR K=-32 TO 31
180         XD=XA*32
190         FOR Y=0 TO 31 STEP 4
200           VPOKE (YD*256)+XD+0,(Y+0)*8+(K AND 7)
210           VPOKE (YD*256)+XD+1,(Y+1)*8+(K AND 56)/8
220           VPOKE (YD*256)+XD+2,(Y+2)*8+(J AND 7)
230           VPOKE (YD*256)+XD+3,(Y+3)*8+(J AND 56)/8
240           XD=XD+4
250         NEXT Y
260         YD=YD+1
270       NEXT K
280       J=J+1
290     NEXT XA
300   NEXT YA
310 NEXT I
320 IF INKEY$="" THEN 330
330 SET PAGE 1,0
340 VDP(9)=VDP(9) OR &B00000010:M=1
350 T=STICK(0)
360 X=X-(T>1 AND T<5)*M+(T>5 AND T<9)*M
370 IF X>511 THEN X=0
380 IF X<0 THEN X=511
390 Y=Y-(T>3 AND T<7)*M+((T=1 OR T=2)OR T=8)*M
400 IF Y>255 THEN Y=0
410 IF Y<0 THEN Y=255
420 SET SCROLL X,Y,0,1
430 GOTO 350
```

リスト2.5　画面左端8ドットのマスク

左8ドットをマスクした場合

```
100 SCREEN 5:CLS
110 FOR X=0 TO 255 STEP 16
120   LINE (X,0)-(X+15,211),X/16,BF
130 NEXT X
140 FOR X=0 TO 255
150   SET SCROLL X,0,1,0
160   FOR I=1 TO 300:NEXT I
170 NEXT X
180 GOTO 140
```

左8ドットをマスクしなかった場合

```
100 SCREEN 5:CLS
110 FOR X=0 TO 255 STEP 16
120   LINE (X,0)-(X+15,211),X/16,BF
130 NEXT X
140 FOR X=0 TO 255
150   SET SCROLL X,0,0,0
160   FOR I=1 TO 300:NEXT I
170 NEXT X
180 GOTO 140
```

PUT　KANJI

●書式●　PUT　KANJI [(<X>, <Y>)], <漢字コード>
　　[, <カラーコード>] [, <ロジカルオペレーション>]
　　[, <モード>]

SCREEN5からSCREEN12で漢字を表示します.

漢字コードがJIS第2水準まで指定できるようになっています. ただし, 表示するためには, JIS第2水準ROMが必要となります. そのほかは, Version 2.0と同じです.

リスト2.6　第2水準漢字表を作る

```
100 '第二水準漢字を表示する
110 SCREEN 7,,,,,3:SET PAGE 1,0
120 OPEN "GRP:" FOR OUTPUT AS #1
130 K=&H50
140 SET PAGE 1,0:CLS
150 SET PAGE 1,1:CLS
160 FOR I=0 TO 2
170   Y=64*I
180   PRESET (40,Y):SET PAGE 1,0
190   PRINT #1," 0 1 2 3 4 5 6 7 8 9 A B C D E F"
200   PRESET (40,Y):SET PAGE 1,1
210   PRINT #1," 0 1 2 3 4 5 6 7 8 9 A B C D E F"
220   FOR KL=2 TO 7
230     PRESET (0,Y+(8*(KL-1))):SET PAGE 1,0
240     PRINT #1,HEX$((K+I)*256+KL*16)
250     PRESET (0,Y+(8*(KL-1))):SET PAGE 1,1
260     PRINT #1,HEX$((K+I)*256+KL*16)
270     FOR X=0 TO 15
280       SET PAGE 1,0
290       PUT KANJI(40+16*X,Y+(8*(KL-1))),(K+I)*256+KL*16+X,15,,1
300       SET PAGE 1,1
310       PUT KANJI(40+16*X,Y+(8*(KL-1))),(K+I)*256+KL*16+X,15,,2
320     NEXT X
330   NEXT KL
340 NEXT I
350 IF INKEY$="" THEN 350
360 K=K+3:IF K<>&H74 THEN 140
370 END
```

VDP

●書式●　VDP(<レジスタ番号>)

VDPのレジスタへデータを書き込んだり，レジスタからデータを読みだしたりします．

この命令は，V9958で増設されたレジスタに対応しています．新設されたレジスタは25から28です．新設レジスタの機能についての詳しいことは次の節で解説します．

指定できる<レジスタ番号>は-9から-1，0から28および33から47です．-9から-1がそれぞれVDPのステータスレジスタ9から1へ，0から7がレジスタ0から7へ，8がステータスレジスタ0へ，9から28がレジスタ8から27へ，33から47がレジスタ32から46へ対応しています．

以下に使い方の例を示します．

```
VDP(26)=VDP(26) OR &B00001000
```

このように指定すると，画面モードがいかなる場合でも，VRAM内のデータをYJKのデータとして出力するようになります．また，次のように指定すると，スプライトの表示を強制的に禁止することができます．

```
VDP(9)=VDP(9) OR &B00000010
```

そのほか影響のあるコマンド

SCREEN10から12のYJKモードでは，グラフィック関係のコマンドは基本的には，SCREEN8と同じ動作をします．ただし，COPYコマンドなどを使うときには，YJKデータの4ドットの区切りに注意して行わなければなりません．

3 グラフィック機能パワフル活用

新VDP　V9958の機能

MSX2＋用に開発されたV9958は，V9938の機能に加え，YJK方式による自然画表示，水平スクロールなどの機能が組み込まれ，また，VDPコマンド実行についての機能の強化がはかられています．しかしその陰で，削られた機能も存在します．ここでは，このV9958の機能について，V9938と比較しながらより詳しく見ていくことにします．

まずはじめに，V9958で追加された機能と削除された機能をあげておきます．

V9958ではV9938に比べて，以下の機能が追加されています．

- YJK方式による自然画表示モードの採用
- 水平方向へのハードウエアスクロール
- VDPのVRAMアクセス時のCPUのポートアクセスに対するウエイト機能
- VDPコマンドをすべてのスクリーンモードで使用可能

削られた機能は以下の2つです．

- コンポジットビデオ出力
- マウス／ライトペンインターフェイス

■ V9958で新設されたレジスタ

V9958で新設されたレジスタは以下のとおりです．もちろん，これらのレジスタはVDPコマンドで操作することができます．

R＃25　VDP(26)　モードレジスタ#5
R＃26　VDP(27)　画面の水平スクロール①
R＃27　VDP(28)　画面の水平スクロール②

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2
R#26	0	0	H08	H07	H06	H05	H04	H03
R#27	0	0	0	0	0	H02	H01	H00

表 2.4　V9958 で新設されたレジスタ

また，V9958 であることを示すために，ステータスレジスタ 1 のなかの ID 番号部分を 2 に変更しています．

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
S#1	FL	LPS	0	0	0	1	0	FH

ID#

ID#＝00001：V9938
＝00010：V9958

表 2.5　V9958 のステータスレジスタ

マウス／ライトペンインターフェース機能削除のため，以下に示すレジスタのビットが意味を持たなくなります．ただし，MSX2 でも使われていない機能なので，とくに問題はないでしょう．これらのレジスタに値を書き込むときは，これらのビットにかならず 0 を書き込んでください．

R＃0　ビット5　IE2　：ライトペンによる割り込み可能
R＃8　ビット7　MS　：マウス使用可能
　　　ビット6　LP　：ライトペン使用可能
S＃1　ビット7　FL　：ライトペン光検出フラグ
　　　ビット6　LPS　：ライトペンスイッチ，マウススイッチ

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
R#0	0	DG	IE2	IE1	M5	M4	M3	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW
S#1	FL	LPS	←―― ID# ――→					FH

表 2.6 V9938 から削除された機能に関するレジスタ

■ 各レジスタの設定

水平スクロール

R#26, 27 の値を変えることによって，画面の水平表示位置を設定することができます．

R#26 は，設定値の分だけ画面を左方向へ移動します．単位は SCREEN6, 7 では 16 ドット単位，それ以外のモードでは 8 ドット単位です．値の範囲は 0 から 63 です．

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
R#26	0	0	H08	H07	H06	H05	H04	H03

表 2.7 水平スクロールに関するレジスタ①

R#27 は，設定値の分だけ画面を右方向へ移動します．単位は SCREEN6, 7 では 2 ドット単位，それ以外のモードでは 1 ドット単位です．値の範囲は 0 から 7 です．

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
R#27	0	0	0	0	0	H02	H01	H00

表 2.8 水平スクロールに関するレジスタ②

R#26,27 を組み合わせて指定することによって，1 ドット単位で水平スクロールをさせることができます(SCREEN6,7 では 2 ドット単位)．

R#25 のビット 1 を指定することによって，スクロール時に画面左側 8

ドット(SCREEN6,7は16ドット)を表示するかしないかを設定することができます．表示を消したときはボーダーカラーが出力されます．

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

MSK=0：画面左側8ドットをマスクし，ボーダーカラーを出力する
=1：画面左側8ドットはマスクされない
(SCREEN6, 7は16ドット)
SP2=0：水平方向画面サイズを1ページとする
=1：水平方向画面サイズを2ページとする

表2.9　マスクと水平方向画面サイズに関するレジスタ

R#25のビット0を指定することによって，水平方向スクロールの画面サイズを変更することができます．

ウエイト機能

R#25のビット2を1にするとV9958のVRAMアクセスが完了するまで，CPUのV9958へのポートアクセスに対してウエイトをかけます．ポートアクセスのみにウエイトがかかるという点に注意してください．

コマンド機能

R#25のビット6を1にすると，すべてのスクリーンモードで，VDPコマンドを使用することができるようになります．SCREEN5～8以外のモードでは，SCREEN8モードとして動作します．ですから，X，Y座標はSCREEN8モードの座標系に従って設定してください．

YJK方式データの表示機能

R#25のビット3を1にすると，画面モードがいかなる状態であっても，VRAM内のデータをYJK方式のデータとみなし，これをRGB信号(RGB各5ビット)に変換して表示します．またこのモードでは，スプライトの表示色にパレットを使うことができます．

このとき，R#25のビット4が1ならば，VRAM内のデータをアトリビュート付きのYJKデータとして扱います．

活用サンプルプログラム①
レイトレーシングプログラム

ここでは，アトリビュートなしYJKモードの1万9千色を使ったレイトレーシングプログラムを紹介します．

このプログラムでは，色を従来どおりRGBの値として計算しています．これは，YJKよりもRGBの方が色の合成を計算するのには優れているからです．そして，計算されたRGBの値をYJKの値に変換する特別な式を使ってYJKデータを作っています．

RGBからYJKへの変換式を使えば，こういったRGBで計算した方が都合のよいものやRGBで存在しているデータをYJKに変換して表示することができます．

■ RGBからYJKへの変換式

レイトレーシングプログラムの使い方を説明するまえに，RGB→YJK変換の式を紹介しましょう．

まず，となり合う4ドットのRGBデータをそれぞれ，

Ri，Bi，Gi　(i=0〜3)

とします．これに対して，それぞれ以下の計算をしてIi，Yi，Ji，Kiを計算します．

$$Ii = \frac{1}{4}Ri + \frac{1}{2}Gi + \frac{1}{8}Bi \qquad -①$$

$$Y'i = \frac{1}{4}Ri + \frac{1}{8}Gi + \frac{1}{2}Bi \qquad -②$$

$$Ji = Ri - Y'i \qquad -③$$

$$Ki = Gi - Y'i \qquad -④$$

①の式はRGBから実際の人間の視覚感度に近い輝度を算出するための式で，この計算式によって得られる値をIi(INTENSITY)とします．②式のY'iは9958でのいわゆる輝度情報に当たる値です．

これらの値をもとに,

$$J = (J0+J1+J2+J3) \div 4 \qquad -⑤$$
$$K = (K0+K1+K2+K3) \div 4 \qquad -⑥$$
$$Yi = (32Ii-6J-15K) \div 29 \qquad -⑦$$

を計算します. この結果得られた値が, 4 ドットの RGB データに対する YJK データとなります. ⑦式で, ①式で得た人間に視覚感度に近い値を使って, ⑤⑥式で平均化された JK に対して Yi の値を補正して, 人間の視覚感度が保存されるようにしています.

これを YJK のデータ形式に従って, VRAM に書き込みます. ただし, この計算式では, RGB のデータはそれぞれ, 0≦Ri, Bi, Gi≦31 の範囲にあることを仮定しています. また, 計算した結果, J, K が−32〜31 をはみだしたら, −32 または 31 に置き換えます. 同様に Yi が 0〜31 をはみ出したら 0 または 31 に置き換えます. 演算途中では, 桁落ちが起こらないよう, 充分注意して計算する必要があります.

この計算式を使えばかなりきれいに RGB データを YJK データに変換することができます. しかし, YJK の特徴から, 色の変化の激しいところや, 細かく色が変化しているところは, どうしてもにじんでしまいますが, これは仕方がありません.

このプログラム中では, RGB を各 5 ビットのデータとして計算しています. そして, 1540 行から 1730 行でこの変換の処理を行い VRAM へデータを書き込んでいます.

■ 物体や光のデータの与え方

このレイトレーシングプログラムはDATA文でパラメータを与えることによって色々な図形を描くことができます．DATAは5000行以上のところから始めるようにします．そして行番号の若い方から以下のようにデータを指定していきます．

5000 DATA	EX, EY, EZ	視点のある所
5010 DATA	PX, PY, PZ	見つめている場所
5020 DATA	LX, LY, LZ	光源のある所
5030 DATA	ZOOM	拡大係数
5100 DATA	NUMOBJ	物体の数
5110 DATA	OX, OY, OZ	物体の中心の位置
5120 DATA	A, B, C	大きさ係数
5130 DATA	TY, SH	種別，シェーデイング番号

3行を1組とした上記のデータを必要な物体の数だけ(NUMOBJで指定した数だけ)書いていきます．

5200 DATA	NUMSHD	シェーデイングの数
5210 DATA	R, G, B	色係数
5220 DATA	AM, DI, MI	環境光，拡散光，鏡面係数
5230 DATA	SP, ME	表面反射光の強さ，表面の磨かれ具合い・鏡面度

3行を1組とした上記のデータを必要なシェーディングの数だけ(NUM-SHDで指定した数だけ)書いていきます．

視点のある位置，見つめている場所，光源のある位置をXYZで指定します．この3次元空間は，一般的に右手系といわれる座標系です．

拡大係数は，XYZで指定した一目盛りを実際にどのくらいの大きさで描くかを示していします．

種別を指定することによって，その物体の形を決めます．指定できる物体の種類は直方体と球です．種別ごとの大きさ係数の指定の仕方は次のようになっています．

TY = 0のときは直方体を描きます．このとき大きさ係数は，

A: X軸の広がり
B: Y軸の広がり
C: Z軸の広がり

というように，図形の中心から図形の辺までの距離をX，Y，Zそれぞれの方向について指定します．

TY = 1のときは球体を描きます．このとき大きさ係数は，

A: X軸の半径の2乗の逆数（cf. A = 1/(Rx * Rx)）
B: Y軸の半径の2乗の逆数
C: Z軸の半径の2乗の逆数

になります．XYZの値を同じにすれば，完全な球になります．またバラバラの値を指定すれば，フットボール状の球を描くことも可能です．

シェーディング番号は次に説明するデータのうち，どのデータを使用するのか番号で指定します．

シェーディングとは，その物体の表面に色を付けるためのデータです．3行を1組として指定していきます．最初のシェーディングデータから1組ごとに順番に0からシェーディング番号が付けられていきます．これを，図形データのシェーディング番号のところに指定するわけです．

鏡面係数とは，物体の表面の状態を表し，数値が大きいほど鏡のようによく光を反射します．1.0のときは，完全な鏡面になります．

色係数は，その物体の色をRGB3値の割合で設定します．環境光は，まわりの物体の照り返しや空気による散乱光の強さを指定します．拡散光は，光源からやって来る光が物体の表面で乱反射されて目にはいる光の強さを指定します．表面反射光の強さは，光源からの光が反射調節されて目にはいってくる光の強さを指定します．鏡面度は，その物体の面がどのぐらい磨きあげられているか（でこぼこがあるかないか）を指定します．

視点や物体の位置を表すXYZの値の範囲は，任意に設定してよいのですが，あまり大きな値を設定すると計算誤差から図形が歪んでしまう可能性があるので，できれば0から10程度の値にした方がよいでしょう．

その他の，色係数やシェーディングの係数はすべて0.0から1.0の範囲で設定しますが．鏡面度だけは0から8位までの整数で指定します．

シェーディング番号に-1を指定すると，シェーディング#0，#1によるチェック模様を形成します．このとき物体の種類は球，直方体のどちらを指定しても大丈夫ですが，球の場合はあまりきれいに表示できません．

■ 使用の際の注意

すべての計算が終わって，絵を描き終るとBASICのコマンドモードに戻ってきますが，画像データはVRAMのページ1に描かれていますので，まだデータは残っています．あわてずにBASICで，

```
10 SCREEN 12:SET PAGE 1,1
20 IF INKEY$="" THEN GOTO 20
```

を実行すれば完成した絵を心行くまで鑑賞することができます．

できた絵をディスク等に保存したいときは，BASICで，

```
SCREEN 12 : SET PAGE 1,1 : BSAVE"FILESNAME",0,&HD400,S
```

とすれば保存できます．

このプログラムでは，SCREEN8の画像を作ることもできます．SCREEN8の絵を書くときは，1010行のリマーク記号を削除し，1480行を削除してください．

データを作るとき，毎回まともに実行していたのでは時間がかかりすぎるので，1ピクセルを4×4ドットで表示して短時間で表示画像の概要を見ることのできるモードを用意しています．このモードを使用するには1180行のMD=0をMD=1に書き変えてください．

絵を描くためにかかる時間は，ベーしっ君を使った場合，図形3つで，3時間程度かかります．BASICの場合は，同じ条件で，16時間ぐらいかかります．図形が増えていけば，どんどん時間がかかるようになっていきます．ベーしっ君を持っていない人は，図形の数を4から5くらいに設定したほうがよいでしょう．

最後に，サンプルとしてデータの書き方の例をあげておきます．

```
5000 ' data
5010 DATA  20, 40, 20    EX,EY,EZ    視点のある所
5020 DATA   0,  0,  0    PX,PY,PZ    見つめている場所
5030 DATA  -8,  9, -3    LX,LY,LZ    光源のある所
5040 DATA   6            ZOOM        拡大係数
5050 DATA   4            NUMOBJ      物体の数
5060 DATA  -2,0, -2      OX,OY,OZ    物体の中心の位置
5070 DATA  .2,.2, .2     A, B, C     大きさ係数
5080 DATA   1,  0        TY,SH       種別，シェーデイング番号
5090 DATA   2,  0,  2    OX,OY,OZ    物体の中心の位置
5100 DATA  .2, .2, .2    A, B, C     大きさ係数
5110 DATA   1,  1        TY,SH       種別，シェーデイング番号
5120 DATA  -2,  0,  2    OX,OY,OZ    物体の中心の位置
5130 DATA  .2, .2, .2    A, B, C     大きさ係数
5140 DATA   1,  2        TY,SH       種別，シェーデイング番号
5150 DATA   2,  0, -2    OX,OY,OZ    物体の中心の位置
5160 DATA  .2, .2, .2    A, B, C     大きさ係数
5170 DATA   1,  3        TY,SH       種別，シェーデイング番号
5180 DATA   4            NUMSHD      シェーデイングの数
5190 DATA  .4, .8, .6    R, G, B     色係数(シェーデイング#0)
5200 DATA  .3, .6,  0    AM,DI,MI    環境光，拡散光，鏡面係数
5210 DATA   0,  0        SP,ME       表面反射光の強さ，鏡面度
5220 DATA  .4, .8, .6    R, G, B     色係数(シェーデイング#1)
5230 DATA  .3, .6,  0    AM,DI,MI    環境光，拡散光，鏡面係数
5240 DATA  .6,  2        SP,ME       表面反射光の強さ，鏡面度
5250 DATA  .4, .8, .6    R, G, B     色係数(シェーデイング#2)
5260 DATA  .3, .6,  0    AM,DI,MI    環境光，拡散光，鏡面係数
5270 DATA  .6,  4        SP,ME       表面反射光の強さ，鏡面度
5280 DATA  .4, .8, .6    R, G, B     色係数(シェーデイング#3)
5290 DATA  .3, .6,  0    AM,DI,MI    環境光，拡散光，鏡面係数
5300 DATA  .6,  6        SP,ME       表面反射光の強さ，鏡面度
```

リスト 2.7 レイトレーシングプログラム TRACE.BAS

```
1000 ' screen init
1010 'SCREEN 8,0:SET PAGE 1,1
1020 SCREEN 12:SET PAGE 1,1
1030 CLS
1040 '_TURBO ON
1050 ' initialize
1060 DIM T(9),V(15),O(19,7),S(19,7),RI(3),GI(3),BI(3),I(3),Z(3),Y(3)
1070 FOR I=0 TO 9:READ T(I):NEXT I
1080 V(0)=T(0):V(1)=T(1):V(2)=T(2)
1090 V(9)=T(0)-T(3):V(10)=T(1)-T(4):V(11)=T(2)-T(5)
1100 V=SQR(V(9)*V(9)+V(10)*V(10)+V(11)*V(11))
1110 V(9)=V(9)/V:V(10)=V(10)/V:V(11)=V(11)/V
1120 V(6)=-V(9)*V(10):V(7)=1-V(10)*V(10):V(8)=-V(11)*V(10)
1130 V(3)=-(V(10)*V(8)-V(11)*V(7)):V(4)=-(V(11)*V(6)-V(9)*V(8))
1140 V(5)=-(V(9)*V(7)-V(10)*V(6)):V(15)=T(9)
1150 V(12)=T(6):V(13)=T(7):V(14)=T(8)
1160 READ MO:FOR I=0 TO MO-1:FOR J=0 TO 7:READ O(I,J):NEXT J,I
1170 READ MS:FOR I=0 TO MS-1:FOR J=0 TO 7:READ S(I,J):NEXT J,I
1180 MA=1000:MI=1E-03:MD=0:PT=4
1190 FOR I=1 TO 4
1200 V=SQR(V(I*3+0)*V(I*3+0)+V(I*3+1)*V(I*3+1)+V(I*3+2)*V(I*3+2))
1210 V(I*3+0)=V(I*3+0)/V:V(I*3+1)=V(I*3+1)/V:V(I*3+2)=V(I*3+2)/V
1220 NEXT I
1230 ' trace
1240 FOR SY=0 TO 211 STEP 4:FOR SX=0 TO 255 STEP 4:XD=0:YD=0
1250 CX=V(0):CY=V(1):CZ=V(2)
1260 VX=V(3)*(SX+XD-128)/99+V(6)*(106-SY-YD)/99-V(9)*V(15)
1270 VY=V(4)*(SX+XD-128)/99+V(7)*(106-SY-YD)/99-V(10)*V(15)
1280 VZ=V(5)*(SX+XD-128)/99+V(8)*(106-SY-YD)/99-V(11)*V(15)
1290 V=SQR(VX*VX+VY*VY+VZ*VZ)
1300 VX=VX/V:VY=VY/V:VZ=VZ/V
1310 CR=0:CG=0:CB=0:RN=0:RF=1
1320 GOSUB 1760
1330 IF CR>=1 THEN CR=.99
1340 IF CG>=1 THEN CG=.99
1350 IF CB>=1 THEN CB=.99
1360 IF MD THEN 1420
1370 RI(XD)=CR:GI(XD)=CG:BI(XD)=CB
1380 XD=XD+1:IF XD<4 THEN 1250
1390 GOSUB 1470:XD=0
1400 YD=YD+1:IF YD<4 THEN 1250
1410 GOTO 1440
```

```
1420 FOR XD=0 TO 3:RI(XD)=CR:GI(XD)=CG:BI(XD)=CB:NEXT XD
1430 FOR YD=0 TO 3:GOSUB 1470:NEXT YD
1440 IF STRIG(0) THEN GOTO 1460
1450 NEXT SX,SY
1460 END
1470 ' draw
1480 GOTO 1560:'IF YOU WONT TO SEE SCREEN8 PICTUER THEN DELETE THIS LI
NE
1490 'MAKE RGB DATA FOR SCREEN8 AND WRITE IT ON THE SCREEN
1500 FOR XD=0 TO 3
1510 CC=INT(GI(XD)*8)*32+INT(RI(XD)*8)*4+INT(BI(XD)*4)
1520 PSET(SX+XD,SY+YD),CC
1530 NEXT XD
1540 RETURN
1550 'MAKE YJK DATA FOR SCREEN12 AND WRITE IT ON THE SCREEN
1560 FOR XD=0 TO 3
1570 RI(XD)=INT(RI(XD)*32):GI(XD)=INT(GI(XD)*32):BI(XD)=INT(BI(XD)*32)
1580 NEXT XD
1590 FOR XD=0 TO 3:I(XD)=RI(XD)/4+GI(XD)/2+BI(XD)/8:NEXT XD
1600 FOR XD=0 TO 3:Z(XD)=RI(XD)/4+GI(XD)/8+BI(XD)/2:NEXT XD
1610 J=INT((RI(0)+RI(1)+RI(2)+RI(3)-Z(0)-Z(1)-Z(2)-Z(3))/4)
1620 IF J<-32 THEN J=-32
1630 IF J> 31 THEN J= 31
1640 K=INT((GI(0)+GI(1)+GI(2)+GI(3)-Z(0)-Z(1)-Z(2)-Z(3))/4)
1650 IF K<-32 THEN K=-32
1660 IF K> 31 THEN K= 31
1670 FOR XD=0 TO 3:Y(XD)=INT((32*I(XD)-6*J-15*K)/29)
1680 IF Y(XD)>31 THEN Y(XD)=31
1690 IF Y(XD)<0  THEN Y(XD)=0
1700 NEXT XD
1710 PSET(SX+0,SY+YD),Y(0)*8+(K AND 7)
1720 PSET(SX+1,SY+YD),Y(1)*8+(K AND 56)/8
1730 PSET(SX+2,SY+YD),Y(2)*8+(J AND 7)
1740 PSET(SX+3,SY+YD),Y(3)*8+(J AND 56)/8
1750 RETURN
1760 ' pixsel
1770 TT=MA
1780 FOR N=0 TO MO-1
1790 GOSUB 1870
1800 IF TT>T AND T>MI THEN TT=T:TN=N:LX=NX:LY=NY:LZ=NZ
1810 NEXT N
1820 IF TT=MA THEN 1860
1830 CX=CX+TT*VX:CY=CY+TT*VY:CZ=CZ+TT*VZ:N=TN
```

```
1840 GOSUB 2150
1850 IF F=1 THEN GOTO 1760
1860 RETURN
1870 ' cross
1880 RX=CX-O(N,0):RY=CY-O(N,1):RZ=CZ-O(N,2)
1890 A=O(N,3):B=O(N,4):C=O(N,5)
1900 ON O(N,6)+1 GOTO 1920,2030
1910 GOTO 1920
1920 ' box
1930 IF VX=0 THEN T1=MA ELSE IF RX<0 THEN T1=-(RX+A)/VX ELSE T1=-(RX-A
)/VX
1940 IF VY=0 THEN T2=MA ELSE IF RY<0 THEN T2=-(RY+B)/VY ELSE T2=-(RY-B
)/VY
1950 IF VZ=0 THEN T3=MA ELSE IF RZ<0 THEN T3=-(RZ+C)/VZ ELSE T3=-(RZ-C
)/VZ
1960 IF ABS(RY+T1*VY)>B OR ABS(RZ+T1*VZ)>C THEN T1=MA
1970 IF ABS(RZ+T2*VZ)>C OR ABS(RX+T2*VX)>A THEN T2=MA
1980 IF ABS(RX+T3*VX)>A OR ABS(RY+T3*VY)>B THEN T3=MA
1990 IF T1=<T2 AND T1=<T3 THEN T=T1:NX=-VX/ABS(VX):NY=0:NZ=0
2000 IF T2=<T3 AND T2=<T1 THEN T=T2:NY=-VY/ABS(VY):NZ=0:NX=0
2010 IF T3=<T1 AND T3=<T2 THEN T=T3:NZ=-VZ/ABS(VZ):NX=0:NY=0
2020 RETURN
2030 ' ball
2040 AA=VX*VX*A+VY*VY*B+VZ*VZ*C
2050 BB=RX*VX*A+RY*VY*B+RZ*VZ*C
2060 CC=RX*RX*A+RY*RY*B+RZ*RZ*C-1
2070 DD=BB*BB-AA*CC
2080 IF DD<0 THEN T=MA:GOTO 2140
2090 T1=(-BB-SQR(DD))/AA:T2=(-BB+SQR(DD))/AA
2100 IF T1<T2 THEN T=T1 ELSE T=T2
2110 NX=A*(RX+T*VX):NY=B*(RY+T*VY):NZ=C*(RZ+T*VZ)
2120 M=SQR(NX*NX+NY*NY+NZ*NZ)
2130 NX=NX/M:NY=NY/M:NZ=NZ/M
2140 RETURN
2150 ' shade
2160 SH=O(N,7)
2170 IF SH<>-1 THEN 2220
2180 PX=INT(ABS(CX+100)/PT-(CX+100<0))
2190 PY=INT(ABS(CY+100)/PT-(CY+100<0))
2200 PZ=INT(ABS(CZ+100)/PT-(CZ+100<0))
2210 SH=(PX+PY+PZ) MOD 2
2220 SR=S(SH,0):SG=S(SH,1):SB=S(SH,2)
2230 SA=S(SH,3):SD=S(SH,4):SF=S(SH,5)
```

```
2240 SP=S(SH,6):SE=S(SH,7)
2250 JX=V(12)-VX:JY=V(13)-VY:JZ=V(14)-VZ
2260 JN=SQR(JX*JX+JY*JY+JZ*JZ)
2270 SM=(LX*JX+LY*JY+LZ*JZ)/JN
2280 IF SM<0 THEN SM=0
2290 FOR P=1 TO SE:SM=SM*SM:NEXT P
2300 VN=-2*(LX*VX+LY*VY+LZ*VZ)
2310 WX=VX+VN*LX:WY=VY+VN*LY:WZ=VZ+VN*LZ
2320 VX=V(12):VY=V(13):VZ=V(14)
2330 SN=LX*VX+LY*VY+LZ*VZ
2340 IF SN<0 THEN SN=0
2350 FOR N=0 TO MO-1
2360 GOSUB 1870
2370 IF MA>T AND T>MI THEN SN=0:SM=0
2380 NEXT N
2390 CR=CR+(SR*(SA+SD*SN)+SP*SM)*RF
2400 CG=CG+(SG*(SA+SD*SN)+SP*SM)*RF
2410 CB=CB+(SB*(SA+SD*SN)+SP*SM)*RF
2420 IF SF=0 AND RN<4 THEN F=0:GOTO 2450
2430 F=1:RF=RF*SF:RN=RN+1
2440 VX=WX:VY=WY:VZ=WZ
2450 RETURN
5000 ' data
5010 DATA   20,  40,  20
5020 DATA    0,   0,   0
5030 DATA   -8,   9,  -3
5040 DATA    6
5050 DATA    4
5060 DATA   -2,   0,  -2
5070 DATA   .2,  .2,  .2
5080 DATA    1,   0
5090 DATA    2,   0,   2
5100 DATA   .2,  .2,  .2
5110 DATA    1,   1
5120 DATA   -2,   0,   2
5130 DATA   .2,  .2,  .2
5140 DATA    1,   2
5150 DATA    2,   0,  -2
5160 DATA   .2,  .2,  .2
5170 DATA    1,   3
5180 DATA    4
5190 DATA   .4,  .8,  .6
5200 DATA   .3,  .6,   0
```

```
5210 DATA   0,   0
5220 DATA  .4,  .8,  .6
5230 DATA  .3,  .6,   0
5240 DATA  .6,   2
5250 DATA  .4,  .8,  .6
5260 DATA  .3,  .6,   0
5270 DATA  .6,   4
5280 DATA  .4,  .8,  .6
5290 DATA  .3,  .6,   0
5300 DATA  .6,   6
```

活用サンプルプログラム②
YJKモード用グラフィックエディタ

YJKモードで絵を作ったり，編集したりできる簡易なグラフィックエディタを紹介します．

これは，SCREENモードでいうとSCREEN10，11のアトリビュート付きのYJKデータをエディットするためのものです．

このプログラムでは，ポインティングデバイスとしてマウスを使うことができます．もちろんキーボードでも使用可能です．

プログラムを実行すると，矢印の形をしたカーソルが表示されます．また，アイコンなどが描かれたメニューウィンドウも表示されます．このカーソルはマウスを動かすことによって，移動することができます．キーボードで使用する場合はカーソルキーを押すことでカーソルを移動させることができます．この場合，シフトキーをカーソルキーと同時に押すと速く移動させることができます．

操作コマンドの実行は，メニューウィンドウの右側に並んでいるアイコンを，カーソルで指定して，マウスの右側のボタン(キーボードの場合はスペースキー)を押すことによって行われます．

このグラフィックエディタには2つのモードがあります．1つは，アトリビュートが0のデータ，すなわちYJKのデータに対して操作をするYJKモードと，アトリビュートビットを立ててパレットコードを書き込むRGBモードに分けられます．これはBASICでのSCREEN10とSCREEN11の違いと同じです．

それぞれのモードはメニューウィンドウのMDと描かれたアイコンをクリックすると切り替えることができます．

それでは，それぞれのモードでの操作の仕方を説明していきます．

■ YJKモードでの操作

RUNさせた直後はこのモードになっています．このモードではYJKデータに対する操作をすることができます．

YJKでは，JKの値を2バイトにまたがって設定しなければならないので，4ドットを1組として扱っています．YJKモードでの1ピクセルは4ドットにあたります．PSETすると横に細長く4ドットを書き込みます．

メニューウィンドウの左側にYJKのカラーを設定するためウィンドウがあります．いちばん左側の枠には，作った色を登録して置くことができる登録枠があります．最初の状態ではデフォルトの色がすでにセットされています．

操作コマンドを選択するまえに色を設定します．まず，どの登録枠を使うかきめます．使いたい登録枠をカーソルで選んで左のボタンをクリックします．

そして，その右側の棒が並んでいるところで色を決めます．

左からYが4つとJKが並んでいます．それぞれの列でX座標を合わせて，上側の＋または下側の－をクリックすることによって，YJKの値を変更することができます．

4つのYはそれぞれ左から，1，2，3，4ドット目のYデータに順番に対応しています．＋や－のところをクリックして棒を延ばしたり縮めたりして，各YJKの値を決めるわけです．Yの値は，棒をいちばん延ばした状態が31，棒が見えなくなった状態が0に当たり，奇数の値を除いた16段階を指定することができます．J，Kは一番延ばした状態が31，棒が見えなくなった状態が，-32に当たります．各YJKの値を調整して希望の色を作ります．

メニューウィンドウの右側のアイコンをクリックするとそれぞれ操作コマンドが実行されます．色はパレットで指定された色で描きます．

YJKモードで使える操作コマンドはPSET，LINE，BOX，BOXFULL，LOAD，SAVEです．

はじめに説明したとおり，4ドットを1ピクセルをとして扱っていますから，すべてのコマンドで，縦は1ドット，横は4ドットとして書き込んでいきます．

■ RGBモード

RGBモードでは，アトリビュートビットを立てて，指定されたパレットコードを書き込んでいきます．色は，512色中15色を選んで使うことができます．YJKモードと同じように，メニューウィンドウの左側に，パレットの色を変更するためのウィンドウがあります．色を変えたいときは，まず，パレットを選択します．色を変更したいパレットの上にカーソルを移動して左ボタンをクリックします．そのとなりにある棒グラフが並んでいる部分で色を変更します．左からRGBの順に並んでいます．それぞれの列にX座標を合わせて上側の＋や下側にあるーをクリックすると，棒が伸びたり縮んだりして，RGBの値を設定することができます．棒が一番延びた状態が7，見えなくなった状態が0に当たります．各RGBの値を変更して希望の色を作ります．

BDRでPAINTコマンド実行時の境界色の指定をすることができます．境界色にしたいパレットを選んで，BDRと書かれたところを，カーソルで選んでクリックすると，いま選んだ色がそこに表示されます．そして，ペイントしたいパレットを選んで，PAINTコマンドを実行してください．BDRの色で囲まれた部分を塗りつぶします．もちろん同じ色を境界色として選ぶこともできます．

RGBモードで使える操作コマンドは，PSET，LINE，BOX，BOXFULL，CIRCLE1，CIRCLE2，FULL，PAINT，SAVE，LOADです．

■ YJKモード，RGBモード共通の使い方

YJKモード，RGBモードともに操作コマンドのアイコンを指定して実行すると，メニューウィンドウは画面から消えます．そして，Qと描かれたアイコンが残ります．コマンドの実行を終わって，メニューウィンドウを表示させたいときは，Qと描かれたアイコンをカーソルで選んで左のボタンをクリックします．

PSETコマンドは左ボタン(キーボードの場合はスペースキー)を押すと1ピクセル分プロットすることができます．押しながらマウスを移動させれば，曲線を描くこともできます．　LINE，BOX等のコマンド，その他のコマンドでは，左ボタンを押して始点決定で，そのまま押しながら終点位置を決め，ボタンを離すと終点が決定され，ラインやボックスを描きます．

右ボタン(キーボードならばNキー)を押すと，直前に実行したコマンドの結果をキャンセルすることができます．ただしメニューを表示してしまった場合は，もうキャンセルすることはできません．

SAVE，LOADコマンドではいったん画面がクリアーされファイル名の一覧が表示されます．そこで，ファイルネームの入力を要求してくるので，メッセージに従って入力していってください．このエディタで使用する画像データのファイル形式は，BASICのCOPYコマンドの画像データの形式です．ファイル名の拡張子は".PIC"です．

BASICで，

```
10 SCREEN 11(またはSCREEN 10)
20 COPY "TEST.PIC" TO (0,0)
30 IF INKEY$="" THEN 30
```

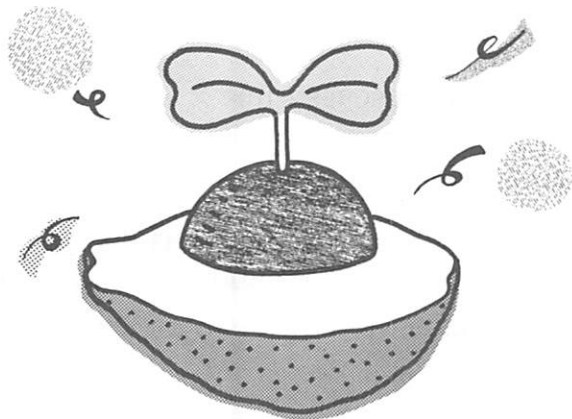
とすれば，読み込むことができます．SCREEN12のアトリビュートなしのYJKモード用のデータをエディットする場合は，いったんSCREEN12でデータを読み込み，SCREEN11または10にモードを変更してセーブし直してください．

たとえば，

```
10 SCREEN 12
20 COPY "TEST.YJK" TO (0,0)
30 SCREEN 11
40 COPY (0,0)-(255,211) TO "TEST.PIC"
```

を実行してください．

こうすることによって，アトリビュートビットがすべてクリアーされ，SCREEN10，11で意味のある画像を表示できるようになります．

YJKの性質から，SCREEN8や7のように1ドットごとに自由に色を付けて絵を描くことは出来ませんが，それでも，YJKで描かれた絵に文字を入れたり，ちょっとYJKのデータを変更したりする程度の用途には使えるでしょう．

リスト 2.8 YJK モード用グラフィックエディタ GRP.BAS

```
1000 'MSX2+ Graphic editor ver.1,0
1010 GOTO 1400
1020 'RGB indicator
1030 FOR I=V+13 TO V+34 STEP8:LINE(I,W+113)-(I+5,W+47),0,BF:NEXT
1040 I=P(C,0):IF I THEN LINE(V+13,W+113)-(V+17,W+113-(I*8)),15,BF
1050 J=P(C,1):IF J THEN LINE(V+21,W+113)-(V+25,W+113-(J*8)),15,BF
1060 L=P(C,2):IF L THEN LINE(V+29,W+113)-(V+33,W+113-(L*8)),15,BF
1070 RETURN
1080 'YJK indicator
1090 FOR I=V+13 TO V+34 STEP 6:LINE(I,W+9)-(I+4,W+113),8,BF:NEXT
1100 LINE(V+39,W+9)-(V+43,W+113),8,BF:LINE(V+47,W+9)-(V+51,W+113),8,BF
1110 X=Y(E,0,0):IF X THEN LINE(V+13,W+113)-(V+17,W+113-(X*4)),&HF8,BF
1120 Y=Y(E,1,0):IF Y THEN LINE(V+19,W+113)-(V+23,W+113-(Y*4)),&HF8,BF
1130 P=Y(E,2,0):IF P THEN LINE(V+25,W+113)-(V+29,W+113-(P*4)),&HF8,BF
1140 Q=Y(E,3,0):IF Q THEN LINE(V+31,W+113)-(V+35,W+113-(Q*4)),&HF8,BF
1150 J=Y(E,0,1):IF J THEN LINE(V+39,W+113)-(V+43,W+113-J),&HF8,BF
1160 K=Y(E,2,1):IF K THEN LINE(V+47,W+113)-(V+51,W+113-K),&HF8,BF
1170 F=(X*16)+8+J MOD 8:RETURN
1180 'color bar
1190 LINE(V+13,W+29)-(V+33,W+34),C,BF:COLOR SPRITE(1)=C:RETURN
1200 'border color bar
1210 LINE(V+13,W+10)-(V+33,W+15),D,BF:RETURN
1220 'YJK color bar
1230 J=W+E*8+2:L=J+6
1240 I=Y(E,3,0)*16+Y(E,0,1)¥8:LINE(V+5,J)-(V+5,L),I:LINE(V+9,J)-(V+9,L
),I
1250 I=Y(E,2,0)*16+Y(E,0,1) MOD 8:LINE(V+4,J)-(V+4,L),I:LINE(V+8,J)-(V
+8,L),I
1260 I=Y(E,1,0)*16+Y(E,2,1)¥8:LINE(V+3,J)-(V+3,L),I:LINE(V+7,J)-(V+7,L
),I
1270 I=Y(E,0,0)*16+Y(E,2,1) MOD 8:LINE(V+2,J)-(V+2,L),I:LINE(V+6,J)-(V
+6,L),I
1280 RETURN
1290 'quitting sprite
1300 I=8-(V>127)*232:PUT SPRITE 2,(I,1),15,8:PUT SPRITE 3,(I,1),4,9:PU
T SPRITE 4,(0,216),0,9:RETURN
1310 'quit (screen mode 10)
1320 P=PEEK(&HCA04):Q=PEEK(&HCA03):IF V>127 THEN 1340
1330 IF Q<10 AND P>6 AND P<17 THEN RETURN 1470 ELSE RETURN
1340 IF Q<10 AND P>238 AND P<249 THEN RETURN 1470 ELSE RETURN
```

```
1350 'quit (screen mode 11)
1360 P=PEEK(&HCA04):Q=PEEK(&HCA03):IF V>128 THEN 1380
1370 IF Q<10 AND P>6 AND P<17 THEN RETURN 2560 ELSE RETURN
1380 IF Q<10 AND P>238 AND P<249 THEN RETURN 2560 ELSE RETURN
1390 'main
1400 CLEAR 200,&HC9FF
1410 MAXFILES=2:DEFINT A-Z:DIM P(15,2),Y(15,3,1),H(15)
1420.SCREEN 10,0,0:SET PAGE 1,1:COLOR 15,0,0:CLS:COLOR SPRITE(0)=14:GO
SUB 4250:GOSUB 4200
1430 GOSUB 4370:DEFUSR=&HCA00:DEFUSR1=&H156:DEFUSR2=&HCA09:DEFUSR3=&HC
A0C:DEFUSR4=&HCA0F
1440 ON ERROR GOTO 4310:OPEN "grp:" FOR OUTPUT AS #1:COLOR SPRITE(1)=1
:C=15:D=15:E=15:GOTO 2560
1450 'Screen 10 mode edit
1460 COPY(0,0)-(255,211),0 TO (0,0),1:GOSUB 3780:GOTO 1480
1470 COPY(0,0)-(255,211),1 TO (0,0),0:GOSUB 3780:PUT SPRITE 2,(0,216)
1480 A=USR(0):X=PEEK(&HCA04)-V:Y=PEEK(&HCA03)-W
1490 IF PEEK(&HCA08) THEN 1460 ELSE IF PEEK(&HCA07)=0 THEN 1480
1500 IF X<2 OR X>68 OR Y<2 OR Y>128 THEN 1480
1510 IF X>34 THEN 1610
1520 IF X>10 THEN 1560
1530 'color
1540 C=(Y-2)¥8:GOSUB 1190:GOSUB 1030:GOTO 1480
1550 'border-color or pallete
1560 IF Y>7 AND Y<16 THEN D=C:GOSUB 1210:GOTO 1480
1570 IF Y>37 AND Y<38+32 THEN I=1 ELSE IF Y>130-32 THEN I=7 ELSE 1480
1580 J=(X-11)¥8:P(C,J)=(P(C,J)+I) MOD 8:GOSUB 1030
1590 COLOR=(C,P(C,0),P(C,1),P(C,2)):GOTO 1480
1600 'i-cion
1610 Z=0:M=((X-36)¥16)*8+(Y-2)¥16:COPY(0,0)-(255,211),0 TO (0,0),1
1620 COPY(0,0)-(255,211),0 TO (0,0),1:GOSUB 1300
1630 'pset,line,box,bf,copy,ld,sv,mode, scrful,paint,cirl,cir2,,,,end
1640 ON M+1 GOTO 1700,1790,1880,1970,2060,3490,3640,2560,2190,2240,229
0,2440,1470,1470,1470,4340
1650 'cancel
1660 COPY(0,0)-(255,211),0 TO (0,0),1:RETURN
1670 'KAKUTEI
1680 COPY(0,0)-(255,211),1 TO (0,0),0:RETURN
1690 'pset
1700 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1320:GOSUB 1660:GOTO 1700
1710 IF PEEK(&HCA07)=0 THEN 1700
1720 GOSUB 1680
1730 X=PEEK(&HCA04):Y=PEEK(&HCA03):PSET(X,Y),C:P=X:Q=Y
```

```
174Ø A=USR(Ø):IF PEEK(&HCAØ7)=Ø THEN 17ØØ
175Ø X=PEEK(&HCAØ4):Y=PEEK(&HCAØ3):LINE(P,Q)-(X,Y),C:P=X:Q=Y
176Ø IF PEEK(&HCAØ7)=1 THEN 174Ø
177Ø GOTO 17ØØ
178Ø 'line
179Ø A=USR(Ø):IF PEEK(&HCAØ8) THEN GOSUB 132Ø:GOSUB 166Ø:GOTO 179Ø
18ØØ IF PEEK(&HCAØ7)=Ø THEN 179Ø
181Ø GOSUB 168Ø
182Ø X=PEEK(&HCAØ4):Y=PEEK(&HCAØ3)
183Ø COPY(Ø,Ø)-(255,211),Ø TO (Ø,Ø),1
184Ø A=USR(Ø):P=PEEK(&HCAØ4):Q=PEEK(&HCAØ3):LINE(X,Y)-(P,Q),C
185Ø IF PEEK(&HCAØ7) THEN 183Ø
186Ø GOTO 179Ø
187Ø 'box
188Ø A=USR(Ø):IF PEEK(&HCAØ8) THEN GOSUB 132Ø:GOSUB 166Ø:GOTO 188Ø
189Ø IF PEEK(&HCAØ7)=Ø THEN 188Ø
19ØØ GOSUB 168Ø
191Ø X=PEEK(&HCAØ4):Y=PEEK(&HCAØ3)
192Ø COPY(Ø,Ø)-(255,211),Ø TO (Ø,Ø),1
193Ø A=USR(Ø):P=PEEK(&HCAØ4):Q=PEEK(&HCAØ3):LINE(X,Y)-(P,Q),C,B
194Ø IF PEEK(&HCAØ7) THEN 192Ø
195Ø GOTO 189Ø
196Ø 'box full
197Ø A=USR(Ø):IF PEEK(&HCAØ8) THEN GOSUB 132Ø:GOSUB 166Ø:GOTO 197Ø
198Ø IF PEEK(&HCAØ7)=Ø THEN 197Ø
199Ø GOSUB 168Ø
2ØØØ X=PEEK(&HCAØ4):Y=PEEK(&HCAØ3)
2Ø1Ø COPY(Ø,Ø)-(255,211),Ø TO (Ø,Ø),1
2Ø2Ø A=USR(Ø):P=PEEK(&HCAØ4):Q=PEEK(&HCAØ3):LINE(X,Y)-(P,Q),C,B
2Ø3Ø IF PEEK(&HCAØ7) THEN 2Ø1Ø
2Ø4Ø LINE(X,Y)-(P,Q),C,BF:GOTO 197Ø
2Ø5Ø 'copy
2Ø6Ø A=USR(Ø):IF PEEK(&HCAØ8) THEN GOSUB 132Ø:GOSUB 166Ø:GOTO 2Ø6Ø
2Ø7Ø IF PEEK(&HCAØ7)=Ø THEN 2Ø6Ø
2Ø8Ø GOSUB 168Ø
2Ø9Ø X=PEEK(&HCAØ4):Y=PEEK(&HCAØ3)
21ØØ COPY(Ø,Ø)-(255,211),Ø TO (Ø,Ø),1
211Ø A=USR(Ø):P=PEEK(&HCAØ4):Q=PEEK(&HCAØ3):LINE(X,Y)-(P,Q),C,B
212Ø IF PEEK(&HCAØ7) THEN 21ØØ
213Ø COPY(Ø,Ø)-(255,211),Ø TO (Ø,Ø),1:LINE(X,Y)-(P,Q),C,B
214Ø A=USR(Ø):IF PEEK(&HCAØ8) THEN GOSUB 132Ø:GOSUB 166Ø:GOTO 2Ø6Ø
215Ø IF PEEK(&HCAØ7)=Ø THEN 214Ø
216Ø I=PEEK(&HCAØ4):J=PEEK(&HCAØ3):GOSUB 166Ø:COPY(X,Y)-(P,Q),1 TO (I,
J),1
```

```
2170 GOTO 2060
2180 'screen full paint
2190 LINE(0,0)-(255,211),C,BF
2200 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1320:GOSUB 1660:GOTO 1480
2210 IF PEEK(&HCA07)=1 THEN 1470
2220 GOTO 2200
2230 'paint
2240 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1320:GOSUB 1660:GOTO 2240
2250 IF PEEK(&HCA07)=0 THEN 2240
2260 GOSUB 1680
2270 X=PEEK(&HCA04):Y=PEEK(&HCA03):PAINT(X,Y),C,D:GOTO 2240
2280 'Circle1(ﾊﾐﾀﾞｼ ﾌｶ)
2290 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1320:GOSUB 1660:GOTO 2290
2300 IF PEEK(&HCA07)=0 THEN 2290
2310 GOSUB 1680
2320 X=PEEK(&HCA04):Y=PEEK(&HCA03)
2330 COPY(0,0)-(255,211),0 TO (0,0),1
2340 A=USR(0):P=PEEK(&HCA04):Q=PEEK(&HCA03):I=ABS(X-P):J=ABS(Y-Q)
2350 IF I<J THEN I=J
2360 IF Y<I THEN I=Y
2370 IF X<I THEN I=X
2380 IF Y+I>211 THEN I=211
2390 IF X+I>255 THEN I=255
2400 CIRCLE(X,Y),I,C
2410 IF PEEK(&HCA07) THEN 2330
2420 GOTO 2290
2430 'Circle2(ﾊﾐﾀﾞｼ ｶ)
2440 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1320:GOSUB 1660:GOTO 2440
2450 IF PEEK(&HCA07)=0 THEN 2440
2460 GOSUB 1680
2470 X=PEEK(&HCA04):Y=PEEK(&HCA03)
2480 COPY(0,0)-(255,211),0 TO (0,0),1
2490 A=USR(0):P=PEEK(&HCA04):Q=PEEK(&HCA03):LINE(X,Y)-(P,Q),C
2500 IF PEEK(&HCA07) THEN 2480
2510 GOSUB 1660:I=ABS(X-P):J=ABS(Y-Q):IF I<J THEN I=J
2520 CIRCLE(X,Y),I,C
2530 GOTO 2440
2540 'Screen11 mode edit
2550 COPY(0,0)-(255,211),0 TO (0,0),1:GOSUB 4000:GOTO 2570
2560 COPY(0,0)-(255,211),1 TO (0,0),0:GOSUB 4000:PUT SPRITE 2,(0,216)
2570 A=USR(0):X=PEEK(&HCA04)-V:Y=PEEK(&HCA03)-W
2580 IF PEEK(&HCA08) THEN 2550 ELSE IF PEEK(&HCA07)=0 THEN 2570
2590 IF X<2 OR X>68+16 OR Y<2 OR Y>128 THEN 2570
```

```
2600 IF X>53 THEN 2760
2610 IF X>10 THEN 2660
2620 'color
2630 LINE(V+1,W+E*8+1)-(V+10,W+E*8+9),8,B:E=(Y-2)¥8
2640 LINE(V+1,W+E*8+1)-(V+10,W+E*8+9),&H88,B:GOSUB 1090:GOTO 2570
2650 'YJK
2660 IF X<18 THEN I=0:J=0:GOTO 2730
2670 IF X<24 THEN I=1:J=0:GOTO 2730
2680 IF X<30 THEN I=2:J=0:GOTO 2730
2690 IF X<36 THEN I=3:J=0:GOTO 2730
2700 IF X<44 THEN I=0:J=1 ELSE I=2:J=1
2710 IF Y>1 AND Y<33 THEN L=1 ELSE IF Y>98 THEN L=63 ELSE 2570
2720 Y(E,I,J)=(Y(E,I,J)+L) MOD 64:GOSUB 1090:GOSUB 1230:GOTO 2570
2730 IF Y>1 AND Y<33 THEN L=1 ELSE IF Y>98 THEN L=15 ELSE 2570
2740 Y(E,I,J)=(Y(E,I,J)+L) MOD 16:GOSUB 1090:GOSUB 1230:GOTO 2570
2750 'i-cion
2760 Z=0:M=((X-54)¥16)*8+(Y-2)¥16:COPY(0,0)-(255,211),0 TO (0,0),1
2770 COPY(0,0)-(255,211),0 TO (0,0),1:GOSUB 1300
2780 POKE &HCA12,Y(E,0,0)*16+Y(E,2,1) MOD 8:POKE &HCA13,Y(E,1,0)*16+Y(
E,2,1)¥8
2790 POKE &HCA14,Y(E,2,0)*16+Y(E,0,1) MOD 8:POKE &HCA15,Y(E,3,0)*16+Y(
E,0,1)¥8
2800 'pset,line,box,bf,copy,ld,sv,mode, scrful,paint,cir1,cir2,,,,end
2810 ON M+1 GOTO 2870,2980,3080,3180,3290,3490,3640,1470,3450,2560,256
0,2560,2560,2560,2560,4340
2820 'cancel
2830 COPY(0,0)-(255,211),0 TO (0,0),1:RETURN
2840 'KAKUTEI
2850 COPY(0,0)-(255,211),1 TO (0,0),0:RETURN
2860 'pset
2870 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 2870
2880 IF PEEK(&HCA07)=0 THEN 2870
2890 GOSUB 2850
2900 X=PEEK(&HCA04):Y=PEEK(&HCA03):P=X:Q=Y:GOSUB 2950
2910 A=USR(0):IF PEEK(&HCA07)=0 THEN 2870
2920 X=PEEK(&HCA04):Y=PEEK(&HCA03):GOSUB 2950:P=X:Q=Y
2930 IF PEEK(&HCA07)=1 THEN 2910
2940 GOTO 2870
2950 IF (VDP(-2) AND 1)=1 THEN 2950
2960 POKE &HCA16,P:POKE &HCA17,Q:POKE &HCA18,X:POKE &HCA19,Y:A=USR2(0)
:RETURN
2970 'line
2980 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 2980
```

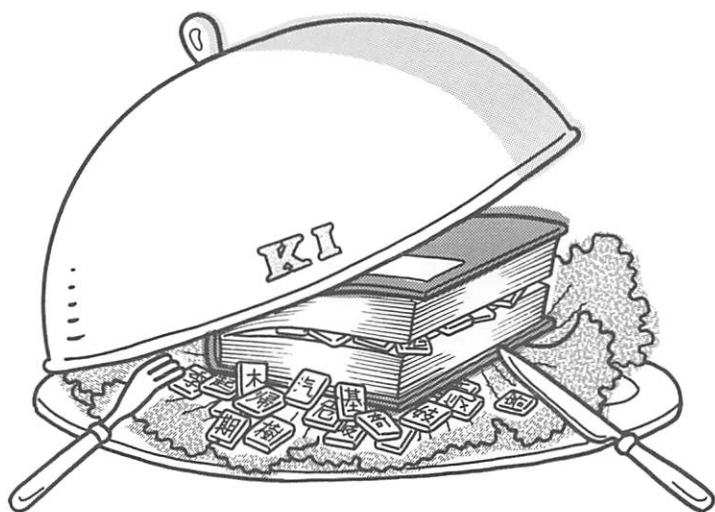
```
2990 IF PEEK(&HCA07)=0 THEN 2980
3000 GOSUB 2850
3010 POKE &HCA16,PEEK(&HCA04):POKE &HCA17,PEEK(&HCA03)
3020 COPY(0,0)-(255,211),0 TO (0,0),1
3030 A=USR(0):POKE &HCA18,PEEK(&HCA04):POKE &HCA19,PEEK(&HCA03)
3040 IF (VDP(-2) AND 1)=1 THEN 3040 ELSE A=USR2(0)
3050 IF PEEK(&HCA07) THEN 3020
3060 GOTO 2980
3070 'box
3080 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 3080
3090 IF PEEK(&HCA07)=0 THEN 3080
3100 GOSUB 2850
3110 POKE &HCA16,PEEK(&HCA04):POKE &HCA17,PEEK(&HCA03)
3120 COPY(0,0)-(255,211),0 TO (0,0),1
3130 A=USR(0):POKE &HCA18,PEEK(&HCA04):POKE &HCA19,PEEK(&HCA03)
3140 IF (VDP(-2) AND 1)=1 THEN 3140 ELSE A=USR3(0)
3150 IF PEEK(&HCA07) THEN 3120
3160 GOTO 3080
3170 'box full
3180 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 3180
3190 IF PEEK(&HCA07)=0 THEN 3180
3200 GOSUB 2850
3210 POKE &HCA16,PEEK(&HCA04):POKE &HCA17,PEEK(&HCA03)
3220 COPY(0,0)-(255,211),0 TO (0,0),1
3230 A=USR(0):POKE &HCA18,PEEK(&HCA04):POKE &HCA19,PEEK(&HCA03)
3240 IF (VDP(-2) AND 1)=1 THEN 3240 ELSE A=USR3(0)
3250 IF PEEK(&HCA07) THEN 3220
3260 IF (VDP(-2) AND 1)=1 THEN 3260 ELSE A=USR4(0)
3270 GOTO 3180
3280 'copy
3290 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 3290
3300 IF PEEK(&HCA07)=0 THEN 3290
3310 GOSUB 2850
3320 POKE &HCA16,PEEK(&HCA04):POKE &HCA17,PEEK(&HCA03)
3330 COPY(0,0)-(255,211),0 TO (0,0),1
3340 A=USR(0):POKE &HCA18,PEEK(&HCA04):POKE &HCA19,PEEK(&HCA03)
3350 IF (VDP(-2) AND 1)=1 THEN 3350 ELSE A=USR3(0)
3360 IF PEEK(&HCA07) THEN 3330
3370 COPY(0,0)-(255,211),0 TO (0,0),1:
3380 IF (VDP(-2) AND 1)=1 THEN 3380 ELSE A=USR3(0)
3390 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 3290
3400 IF PEEK(&HCA07)=0 THEN 3390
3410 I=PEEK(&HCA04):J=PEEK(&HCA03):GOSUB 2830
```

```
3420 COPY(PEEK(&HCA16),PEEK(&HCA17))-(PEEK(&HCA18)+3,PEEK(&HCA19)),0 T
O (I,J),1
3430 GOTO 3290
3440 'screen full paint
3450 POKE &HCA16,0:POKE &HCA17,0:POKE &HCA18,252:POKE &HCA19,211:A=USR
4(0)
3460 A=USR(0):IF PEEK(&HCA08) THEN GOSUB 1360:GOSUB 2830:GOTO 2570
3470 IF PEEK(&HCA07)=1 THEN 2560 ELSE 3460
3480 'load
3490 SET PAGE 0,0:SCREEN 0:COLOR 15,0:CLS
3500 PRINT"LOAD OK(Y/N)?";
3510 A=USR1(0):A$=INPUT$(1):IF A$<>"Y" AND A$<>"y" THEN 3620 ELSE PRIN
T A$
3520 COLOR 8:PRINT"SET A GRAPHIC DISK AND HIT A KEY!":COLOR 15
3530 A=USR1(0):A$=INPUT$(1):PRINT:FILES"*.PIC":PRINT
3540 INPUT"File name:";F$:I=INSTR(F$,"."):IF I THEN F$=LEFT$(F$,I-1)
3550 PRINT:PRINT"Sure(Y/N)?":A$=INPUT$(1):IF A$<>"Y" AND A$<>"y" THEN
3500
3560 SCREEN 10+S
3570 A$=F$+".pic":COPY A$ TO (0,0),1:CLOSE
3580 A$=F$+".rgb":COPY A$ TO H:CLOSE
3590 FOR I=0 TO 15:P(I,0)=H(I)¥100:P(I,1)=H(I)¥10 MOD 10:P(I,2)=H(I) M
OD 10
3600 COLOR=(I,P(I,0),P(I,1),P(I,2)):NEXT
3610 OPEN "grp:" FOR OUTPUT AS #1:GOTO 3760
3620 SCREEN 10+S:COPY(0,0)-(255,211),1 TO (0,0),0:GOTO 3760
3630 'save
3640 SET PAGE 0,0:SCREEN 0:COLOR 15,0:CLS
3650 PRINT"SAVE OK(Y/N)?";
3660 A=USR1(0):A$=INPUT$(1):IF A$<>"Y" AND A$<>"y" THEN 3750 ELSE PRIN
T A$
3670 COLOR 8:PRINT"SET A GRAPHIC DISK AND HIT A KEY!":COLOR 15
3680 A=USR1(0):A$=INPUT$(1):PRINT:FILES"*.PIC":PRINT
3690 INPUT"File name:";F$:I=INSTR(F$,"."):IF I THEN F$=LEFT$(F$,I-1)
3700 PRINT:PRINT"Sure(Y/N)?":A$=INPUT$(1):IF A$<>"Y" AND A$<>"y" THEN
3650
3710 SCREEN 10+S:FOR I=0 TO 15:H(I)=P(I,0)*100+P(I,1)*10+P(I,2):NEXT
3720 A$=F$+".pic":COPY(0,0)-(255,211),1 TO A$:CLOSE
3730 A$=F$+".rgb":COPY H TO A$:CLOSE
3740 OPEN "grp:" FOR OUTPUT AS #1:GOTO 3760
3750 SCREEN 10+S:COPY(0,0)-(255,211),1 TO (0,0),0
3760 ON S+1 GOTO 1470,2560
3770 'screen 10 mode
```

```
3780 SCREEN 10:SET PAGE 1,1:S=0:COLOR 15,0:X=PEEK(&HCA04):Y=PEEK(&HCA0
3)
3790 V=6-(X>127)*168:W=8-(Y>129)*72:LINE(V,W)-(V+69,W+130),0,BF
3800 PSET(V+13,W+115),0:PRINT #1,"RGB"
3810 PSET(V+13,W+1),0:PRINT #1,"BDR":PSET(V+13,W+20),0:PRINT #1,"COL"
3820 PSET(V+21,W+123),0:PRINT #1,"-":PSET(V+21,W+39),0:PRINT #1,"+"
3830 LINE(V,W)-(V+69,W+130),15,B:LINE(V+11,W)-(V+35,W+130),15,B
3840 LINE(V+11,W+46)-(V+35,W+122),15,B
3850 LINE(V+11,W+114)-(V+35,W+114),15
3860 LINE(V+19,W+46)-(V+27,W+122),15,B
3870 LINE(V+11,W+8)-(V+35,W+17),15,B:LINE(V+11,W+19)-(V+35,W+19),15
3880 LINE(V+11,W+27)-(V+35,W+36),15,B:LINE(V+11,W+38)-(V+35,W+38),15
3890 I=0:FOR J=W+2 TO W+128 STEP 8:LINE(V+2,J)-(V+9,J+6),I,BF:I=I+1:NE
XT J
3900 FOR J=W+2 TO W+128 STEP 16:FOR I=V+37 TO V+68 STEP 16
3910 LINE(I,J)-(I+14,J+14),15,BF:NEXT I,J
3920 RESTORE 3970:COLOR 4,15
3930 FOR I=V+38 TO V+68 STEP 16:FOR J=W+6 TO W+128 STEP 16
3940 READ A$:PSET(I,J),0:PRINT #1,A$:NEXT J,I:COLOR 15,0
3950 LINE(V+68,W+1)-(V+68,W+129),0:LINE(V+52,W+1)-(V+52,W+129),0
3960 GOSUB 1030:GOSUB 1190:GOSUB 1210:POKE &HCA1A,0:RETURN
3970 DATA "Ps","Ln","Bx","Bf","Cp","Ld","Sv","Md"
3980 DATA "Fl","Pa","C1","C2","  ","  ","  ","En"
3990 'screen mode 11
4000 SCREEN 11:SET PAGE 1,1:S=1:X=PEEK(&HCA04):Y=PEEK(&HCA03)
4010 V=6-((X>127)*152):W=8-((Y>129)*72)
4020 COLOR 248,8:LINE(V,W)-(V+86,W+130),8,BF
4030 PSET(V+23,W+115),8:PRINT #1,"Y JK"
4040 PSET(V+30,W+123),8:PRINT #1,"-":PSET(V+30,W+1),8:PRINT #1,"+"
4050 LINE(V,W)-(V+87,W+130),,B:LINE(V+11,W)-(V+53,W+130),,B
4060 LINE(V+11,W+8)-(V+53,W+122),,B:LINE(V+11,W+114)-(V+53,W+114)
4070 LINE(V+37,W+8)-(V+45,W+122),,B
4080 F=E:FOR E=0 TO 15:GOSUB 1230:NEXT:E=F
4090 RESTORE 4170:COLOR 248,8
4100 FOR J=W+2 TO W+128 STEP 16:FOR I=V+55 TO V+85 STEP 16
4110 LINE(I,J)-(I+14,J+14),,BF:NEXT I,J:COLOR 72,248
4120 FOR I=V+56 TO V+85 STEP 16:FOR J=W+6 TO W+128 STEP 16
4130 READ A$:PSET(I,J),0:PRINT #1,A$:NEXT J,I:COLOR 248,8
4140 LINE(V+86,W+1)-(V+86,W+129),8:LINE(V+70,W+1)-(V+70,W+129),8
4150 LINE(V+1,W+(E*8)+1)-(V+10,W+(E*8)+9),136,B:GOSUB 1090
4160 GOSUB 1090:GOSUB 1230:POKE &HCA1A,1:RETURN
4170 DATA "Ps","Ln","Bx","Bf","Cp","Ld","Sv","Md"
4180 DATA "Fl","  ","  ","  ","  ","  ","  ","En"
```

```
4190 'RGBﾊﾟﾚｯﾄ
4200 RESTORE 4230:FOR I=0 TO 15:READ J:P(I,0)=J¥100:P(I,1)=(J¥10) MOD
10
4210 P(I,2)=J MOD 10:COLOR=(I,P(I,0),P(I,1),P(I,2)):NEXT
4220 FOR I=0 TO 15:FOR J=0 TO 3:Y(I,J,0)=I:Y(I,J,1)=0:NEXT J,I:RETURN
4230 DATA 0,0,161,373,117,237,511,267,711,733,661,663,141,625,555,777
4240 'sprite pattern
4250 RESTORE 4260:FOR I=&HF000 TO &HF04F:READ A:VPOKE I,A:NEXT:RETURN
4260 DATA 252,144,136,196,162,145,10,4,0,98,112,56,28,14,4,0
4270 DATA 63,9,17,35,69,137,80,32,0,6,14,28,56,112,32,0
4280 DATA 4,10,145,162,196,136,144,252,0,4,14,28,56,112,96,0
4290 DATA 32,80,137,69,35,17,9,63,0,32,112,56,28,14,6,0
4300 DATA 255,199,187,187,171,147,197,255,0,56,68,68,84,108,58,0
4310 'ON ERROR
4320 IF ERR<>53 THEN ON ERROR GOTO 0:COLOR 15,4,7:END
4330 COLOR=NEW:CLOSE:RESUME 3610
4340 ' END
4350 CLOSE:CLS:COLOR 15,4,7:END
4360 'machine language
4370 RESTORE 4380:FOR I=&HCA00 TO &HCDDF:READ K$:K=VAL("&H"+K$):POKE I
,K:NEXT I:RETURN
4380 DATA C3,42,CA,00,00,00,00,00,00,C3,80,CB,C3,09,CD,C3
4390 DATA 73,CD,00,00,00,00,00,00,00,00,00,00,00,00,00,00
4400 DATA 00,00,FF,00,FF,01,00,01,01,01,01,00,01,FF,00,FF
4410 DATA FF,FF,FA,00,FA,06,00,06,06,06,06,00,06,FA,00,FA
4420 DATA FA,FA,F5,C5,D5,E5,DD,E5,FD,E5,AF,CD,D5,00,B7,28
4430 DATA 1D,87,5F,3E,06,CD,41,01,21,30,CA,CB,47,28,03,21
4440 DATA 20,CA,16,00,19,7E,23,66,6F,22,05,CA,18,19,3E,0C
4450 DATA CD,DB,00,3E,0D,CD,DB,00,CB,2F,32,06,CA,3E,0E,CD
4460 DATA DB,00,CB,2F,32,05,CA,3E,01,CD,D8,00,B7,28,02,3E
4470 DATA 01,32,07,CA,3E,03,CD,D8,00,B7,28,02,3E,01,32,08
4480 DATA CA,3E,08,CD,41,01,CB,47,20,05,3E,01,32,07,CA,3E
4490 DATA 04,CD,41,01,CB,5F,20,05,3E,01,32,08,CA,3A,05,CA
4500 DATA 4F,2A,03,CA,5C,AF,47,57,87,CB,79,20,0A,09,7D,FE
4510 DATA D4,38,0F,3E,D3,18,0B,79,ED,44,4F,B7,ED,42,7D,30
4520 DATA 01,AF,32,03,CA,EB,3A,1A,CA,B7,28,05,3A,1B,CA,B5
4530 DATA 6F,3A,06,CA,4F,06,00,CB,79,20,0A,09,7C,B7,7D,28
4540 DATA 0F,3E,FF,18,0B,79,ED,44,4F,B7,ED,42,7D,30,01,AF
4550 DATA 32,04,CA,3A,1A,CA,B7,28,0F,3A,04,CA,5F,E6,FC,32
4560 DATA 04,CA,7B,E6,03,32,1B,CA,2A,03,CA,7D,FE,CB,38,15
4570 DATA D6,09,6F,7C,FE,F7,38,07,D6,08,67,3E,06,18,14,24
4580 DATA 3E,04,C3,53,CB,7C,FE,F7,38,07,D6,08,67,3E,02,18
4590 DATA 02,24,AF,22,77,CB,22,7B,CB,32,79,CB,3C,32,7D,CB
```

```
46ØØ DATA 21,77,CB,11,ØØ,FA,Ø1,Ø8,ØØ,F3,CD,5C,ØØ,FB,FD,E1
461Ø DATA DD,E1,E1,D1,C1,F1,C9,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,Ø1,ØØ,D8
462Ø DATA F5,C5,D5,E5,DD,E5,FD,E5,ED,5B,16,CA,3A,19,CA,92
463Ø DATA CA,A9,CC,3Ø,Ø8,ED,44,21,ØØ,FF,C3,AØ,CB,21,ØØ,Ø1
464Ø DATA 22,2Ø,CA,3C,47,32,1D,CA,3A,18,CA,93,CA,BF,CC,3Ø
465Ø DATA Ø8,ED,44,21,FC,FF,C3,BC,CB,21,Ø4,ØØ,22,1E,CA,CB
466Ø DATA 3F,CB,3F,3C,32,1C,CA,B8,DA,18,CC,67,68,CD,B6,CD
467Ø DATA 7C,B7,28,29,3A,1D,CA,94,FE,Ø2,38,2C,CB,3F,4F,45
468Ø DATA E5,CD,65,CC,E1,4C,45,Ø4,E5,CD,65,CC,E1,3A,1D,CA
469Ø DATA 94,CB,3F,CE,ØØ,4F,45,CD,65,CC,C3,6E,CB,3A,1D,CA
47ØØ DATA 4F,45,CD,65,CC,C3,6E,CB,4F,45,E5,CD,65,CC,E1,4C
471Ø DATA 45,Ø4,CD,65,CC,C3,6E,CB,6Ø,6F,CD,B6,CD,7C,B7,28
472Ø DATA 29,3A,1C,CA,94,FE,Ø2,38,2C,CB,3F,4F,45,E5,CD,87
473Ø DATA CC,E1,4C,45,Ø4,E5,CD,87,CC,E1,3A,1C,CA,94,CB,3F
474Ø DATA CE,ØØ,4F,45,CD,87,CC,C3,6E,CB,3A,1C,CA,4F,45,CD
475Ø DATA 87,CC,C3,6E,CB,4F,45,E5,CD,87,CC,E1,4C,45,Ø4,CD
476Ø DATA 87,CC,C3,6E,CB,C5,C5,21,12,CA,Ø1,Ø4,ØØ,D5,F3,CD
477Ø DATA 5C,ØØ,FB,D1,2A,1E,CA,19,EB,C1,1Ø,EA,2A,2Ø,CA,19
478Ø DATA EB,C1,ØD,C2,65,CC,C9,C5,C5,21,12,CA,Ø1,Ø4,ØØ,D5
479Ø DATA F3,CD,5C,ØØ,FB,D1,2A,2Ø,CA,19,EB,C1,1Ø,EA,2A,1E
48ØØ DATA CA,19,EB,C1,ØD,C2,87,CC,C9,3A,18,CA,6F,93,3Ø,Ø3
481Ø DATA ED,44,5D,CB,3F,CB,3F,3C,47,CD,C5,CC,C3,6E,CB,CD
482Ø DATA F2,CC,C3,6E,CB,EB,F3,CD,71,Ø1,F3,3A,12,CA,CD,E7
483Ø DATA CC,3A,13,CA,CD,E7,CC,3A,14,CA,CD,E7,CC,3A,15,CA
484Ø DATA CD,E7,CC,1Ø,E6,FB,C9,C5,ED,4B,Ø7,ØØ,ED,79,C1,C5
485Ø DATA C1,C9,C5,21,12,CA,Ø1,Ø4,ØØ,D5,F3,CD,5C,ØØ,FB,D1
486Ø DATA 2A,2Ø,CA,19,EB,C1,1Ø,EA,C9,F5,C5,D5,E5,DD,E5,FD
487Ø DATA E5,ED,5B,16,CA,3A,19,CA,92,CA,A9,CC,3Ø,Ø7,ED,44
488Ø DATA 21,ØØ,FF,18,Ø3,21,ØØ,Ø1,22,2Ø,CA,3C,47,32,1D,CA
489Ø DATA 3A,18,CA,6F,93,CA,BF,CC,3Ø,Ø3,ED,44,5D,CB,3F,CB
49ØØ DATA 3F,3C,47,32,1C,CA,D5,CD,C5,CC,D1,3A,1C,CA,47,3A
491Ø DATA 19,CA,57,CD,C5,CC,3A,1D,CA,47,ED,5B,16,CA,CD,F2
492Ø DATA CC,3A,1D,CA,47,3A,17,CA,57,3A,18,CA,5F,CD,F2,CC
493Ø DATA C3,6E,CB,F5,C5,D5,E5,DD,E5,FD,E5,ED,5B,16,CA,2A
494Ø DATA 18,CA,7C,92,CA,A9,CC,3Ø,Ø3,ED,44,54,3C,4F,7D,93
495Ø DATA 28,1A,3Ø,Ø3,ED,44,5D,CB,3F,CB,3F,3C,47,C5,D5,CD
496Ø DATA C5,CC,D1,14,C1,ØD,C2,9D,CD,C3,6E,CB,41,21,ØØ,Ø1
497Ø DATA 22,2Ø,CA,C3,BF,CC,7C,BD,38,14,97,4F,Ø6,Ø8,CB,21
498Ø DATA CB,24,17,BD,38,Ø3,95,CB,C1,1Ø,F3,67,69,C9,2E,ØØ
499Ø DATA C9,CD,79,C9,CD,99,CE,38,22,28,2Ø,CD,99,CE,28,21
```

MSX2+
日本語処理機能

MSX漢字ドライバ

これまでMSXでは，CPUの処理能力や，画面表示能力から，日本語処理が苦手だとされてきました．しかし，MSX-Writeの出現によって，MSXでも十分漢字を扱えることが実証され，その後も日本語対応のアプリケーション・ソフトウェアがいくつか登場しています．ところが，これらのソフトウェアは，それぞれ独自に日本語に対応しているため，日本語処理が各ソフトウェア間でまちまちでした．そのため，ユーザーが直面する部分もソフトウェアごとに異なり，たとえば漢字の入力方法なども，それぞれのソフトウェアで用意された方式に従わなければなりませんでした．

また，ユーザーがプログラムを自作する場合も，日本語処理の部分を1から作っていかねばならず，たいへんな労力を必要としました．とくにBASICではメモリ容量の制限もあるため，普通にプログラムを組んだ場合には，日本語に対応した実用的なアプリケーション・ソフトウェアはとても望めませんでした．

そこで，これらの日本語処理をアプリケーションから切り放して標準化し，システム・ソフトウェアとして組み込み，他のさまざまなアプリケーションから簡単に漢字を利用できるようにしたものがMSX漢字ドライバです．

このMSX漢字ドライバは，VRAM128KまたはVRAM64KのMSX2，MSX2+いずれの機種であっても動作するように設計されています．また，MSX-DOS,MSX-DOS2,BASICのいかなる環境においても動作します．ですから，BASICのプログラムや，DOS上のアプリケーションから漢字の入出力を行うことができます．

この漢字ドライバは，MSXで決められた文字表示ルーチンを使って文字表示を行っている(BIOSコールを使用して文字表示を行っている)プログラムであれば，そのまま利用することができます．BASICのプログラムからは，特殊なことをしていないかぎり，まず問題なく利用できるでしょう．

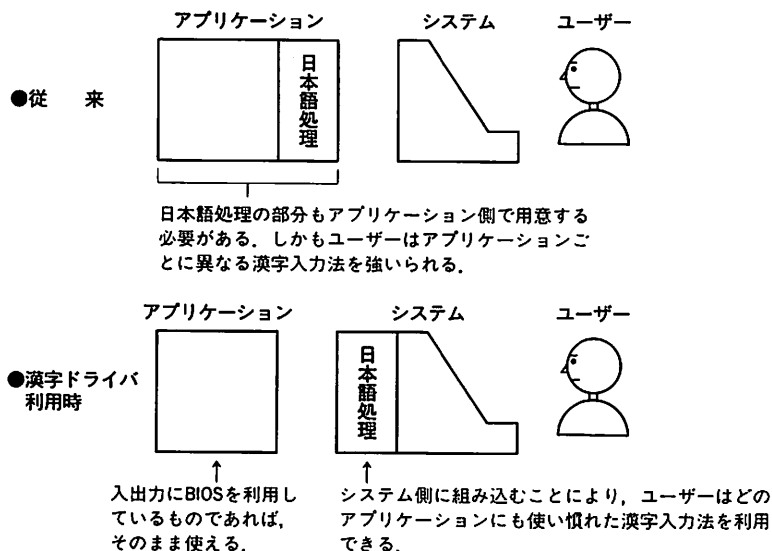



図 3.1　漢字ドライバ概念図

1 バイト文字と 2 バイト文字

ここで，MSX 漢字ドライバを説明する前に，パソコン上で日本語を扱うために必要な知識として，1 バイト文字と 2 バイト文字について述べておきましょう．

■ MSX 内部での文字の取り扱い

コンピュータの内部では，すべての情報が 2 進数で表現されているということはよくご存じだと思います．この 2 進数で表現できる情報量の単位をビットといいます．2 進数 1 桁で表わせる情報量が 1 ビット，2 桁で表わせる情報量が 2 ビットです．ちなみに 1 ビットは 0 と 1 の 2 つのコード，つまり 2 種類の状態を，2 ビットでは 0，1，10，11 と 4 つのコード，つまり 4 種類の状態を表わすことができます．一般に，N ビットで表わせる情報量は 2 の N 乗ということになります．

MSXでは，基本的に8ビット単位で情報を扱います．ですから，MSXで文字を処理するためには，この8ビットの各コードにアルファベットや数字，カタカナなどの各文字を対応させて扱わなければなりません．具体的には，“1000001”のコードには“A”が，“1000010”のコードには“B”が割り当てられているといった具合いです．

ところで，コンピュータの世界では8ビットをさらに1つの単位とし，バイトと呼んでいるため，これらの8ビットで表現される文字を1バイト文字と呼んでいます(普通1バイトのコードは2桁の16進数で表記されます)．

1バイト(8ビット)文字コード

01000001B ⇨ 41H ⇨ A

2進コード　16進コード　対応する文字

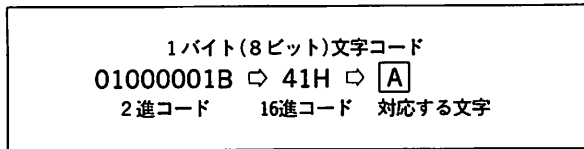



図3.2　1バイト(8ビット)文字コード

このように1バイトのコードと文字を対応させるものには，いくつか規格があり，代表的なものに米国のASCIIコード，日本のJISコードなどがあります．

MSXで採用している文字コードは，JISに準拠した拡張文字コードと呼ばれるもので，JISコードに比べ，グラフィック・キャラクタやひらがなの部分が拡張されています(表3.1)．

このうち，一部のグラフィック・キャラクタと漢字については，1バイトのコードに収まっていません．たとえば，月という文字は1Hと41Hの連続で表わされます．つまり2バイトで1文字を表現しています．このような文字を2バイト文字と呼びます．

1バイトの情報量では，最大でも256種類の文字しか表現できません．これでは，数万種類ともいわれる膨大な文字数を誇る漢字を表現するには，あまりにも少なすぎます．そこで既存の1バイト文字との互換性を保ちながら，漢字を表現するために考えられたのが，2バイトを一組にして1つの文字を表現する2バイト文字コードです．これにより，情報量は16ビットになるので，理論的には2の16乗で65536通りの文字を表現できることになります．

下位＼上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	4	5
0	↕コントロールコード			0	@	P	`	p	♠			ー	タ	ミ	た	み		π
1			!	1	A	Q	a	q	♥	あ	。	ア	チ	ム	ち	む	月	┴
2			"	2	B	R	b	r	♣	い	「	イ	ツ	メ	つ	め	火	┬
3			#	3	C	S	c	s	◆	う	」	ウ	テ	モ	て	も	水	┤
4			$	4	D	T	d	t	○	え	、	エ	ト	ヤ	と	や	木	├
5			%	5	E	U	e	u	●	お	・	オ	ナ	ユ	な	ゆ	金	┼
6			&	6	F	V	f	v	を	か	ヲ	カ	ニ	ヨ	に	よ	土	│
7			'	7	G	W	g	w	ぁ	き	ァ	キ	ヌ	ラ	ぬ	ら	日	─
8			(	8	H	X	h	x	ぃ	く	ィ	ク	ネ	リ	ね	り	年	┌
9			)	9	I	Y	i	y	ぅ	け	ゥ	ケ	ノ	ル	の	る	円	┐
A			*	:	J	Z	j	z	ぇ	こ	ェ	コ	ハ	レ	は	れ	時	└
B			+	;	K	[	k	{	ぉ	さ	ォ	サ	ヒ	ロ	ひ	ろ	分	┘
C			,	<	L	¥	l	¦	ゃ	し	ャ	シ	フ	ワ	ふ	わ	秒	╳
D			−	=	M	]	m	}	ゅ	す	ュ	ス	ヘ	ン	へ	ん	百	大
E			.	>	N	^	n	~	ょ	せ	ョ	セ	ホ	゛	ほ		千	中
F			/	?	O	_	o		っ	そ	ッ	ソ	マ	゜	ま	■	万	小

CHR$(&H41)="A"になる.
グラフィックキャラクタコードは最初にCHR$(1)を付ける.
CHR$(1)+CHR$(&H41)="月"

表 3.1　MSX の JIS 準拠拡張文字コード表

■ 2バイト文字の表現

さて，この1バイト文字と2バイト文字がそれぞれ独立した環境にあれば問題ないのですが，それらが同じ環境に混在すると，処理が難しくなります．たとえば，文字列の先頭の1バイトのコードがあるとき，それが1バイト文字コードであるか，それとも2バイト文字コードの1バイト目であるかを判別しなければなりません．

MSX 漢字ドライバで使われている2バイト文字コードは，JIS の漢字コードをもとに作られたシフト JIS コード(より正確には，さらにシフト JIS をもとにした MS 漢字コード)と呼ばれているものです．この文字コードは，もともと16ビット・パソコンの世界で有名な MS-DOS で使われているものです．

シフトJISコードは，JISの1バイトコードでは未定義のコード(81H～9FH，E0H～FEH)を1バイト目に使用することにより，1バイト文字と2バイト文字を識別するようにしています．

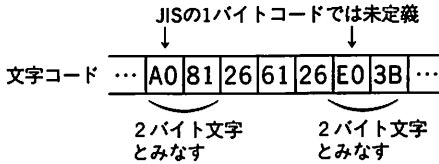



図 3.3　シフト JIS コードのしくみ

MSXでは，シフトJISコードの1バイト目に使用するコードに，ひらがなやグラフィック・キャラクタが定義してあります．したがって，漢字ドライバを使って2バイト文字を利用している場合には，これらのひらがなやグラフィック・キャラクタを使うことはできなくなります．また，MSX独自の2バイト半角の漢字など(1Hではじまるもの)も表示できません．

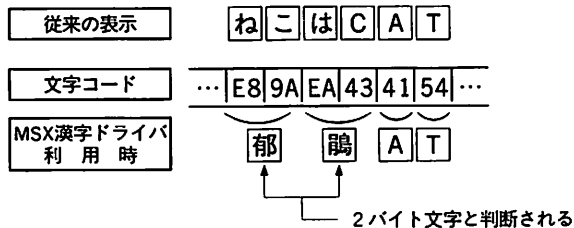



図 3.4　漢字ドライバ利用時の 1 バイト文字のひらがなの扱い

しかし，ひらがなやグラフィック・キャラクタは2バイト文字でも用意されているので，そちらを使えば，まず問題はないでしょう．

MSX漢字ドライバ起動時の画面出力では，1バイト文字と2バイト文字の大きさが異なり，通常1バイト文字は，横幅が2バイト文字の半分のサイズで表示されます．そのため，1バイト文字は半角文字，2バイト文字は全角文

字，あるいは単に半角，全角と呼ばれています．

現在，JIS およびシフト JIS で定義されている2バイト文字には，ひらがな，カタカナ，数字，アルファベット，各種記号および使用頻度高い漢字約3000文字からなる JIS 第1水準，そして第1水準ほどではないけれどやはり使うことの多い漢字を集めた JIS 第2水準という2つのレベルがあります．

MSX 漢字ドライバは，JIS 第2水準までサポートしていますので，第2水準漢字 ROM を装備していれば，これらの文字を画面に出力することができます．

MSX 漢字ドライバの機能

MSX 漢字ドライバは，BIOS と同レベルに位置するシステム・ソフトウェアで，漢字を表示したり入力したりするための色々な処理を，一手に引き受けてくれるプログラムです．漢字ドライバを起動すると漢字ドライバは SCREEN モードをグラフィックモードに変更し漢字フォントを書き込める状態にします．そして，BIOS の文字出力ルーチンに送られたデータを受け取り，その文字が2バイト文字か1バイト文字かを判別して，その文字のフォントをグラフィック画面に書き込むという処理をしています．

BASIC や DOS 側から見れば，今までと同じ BIOS による文字出力をするだけで，漢字の出力が可能になるわけです．また，画面出力だけでなく，漢字プリンタへの出力もサポートしていますので，漢字を含んだプログラムリストをプリンタに出力することもできます．

漢字の入力には，キーボードから入力した文字を漢字に変換してくれる，日本語入力フロントプロセッサはなくてはならない存在です．漢字ドライバを使えば簡単に日本語入力フロントプロセッサを呼び出して，漢字変換を使って文字を入力することが可能になります．

この機能を実現するために，漢字ドライバはキー入力をつねに監視して日本語入力フロントプロセッサの起動の合図を待っています．フロントプロセッサが呼び出されたら，漢字変換用のウィンドウを画面に表示し，キーボードから入力された文字をフロントプロセッサを使って漢字に変換し，プログラムにその変換の結果を渡します．

このように，漢字ドライバは，BIOSの入出力を拡張して，BASICやMSX-DOSの環境から漢字を簡単に利用できるようにしているのです．

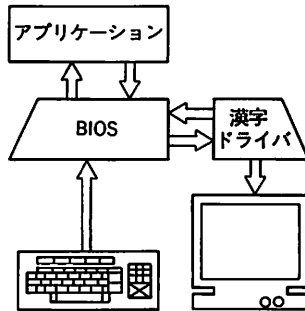



図3.5　MSX漢字ドライバの位置付け

また漢字ドライバは，BASICレベルで2バイト文字と1バイト文字が混ざった文字列を正しく処理できるように作られた，文字列加工用のコマンドをサポートしています．

漢字ドライバで使えるフロントプロセッサは，MSX-WriteやHAL-NOTEなどに登載されているような，MSX-JE規格に添って作られたものです．もし，これらの日本語入力フロントプロセッサがない場合でも，MSX漢字ドライバには標準で単漢字変換システムが内蔵されているので，ちょっと貧弱ではありますが，漢字を入力することは可能です．もちろん，MSX-WriteやMSX-Write II，HAL-NOTEなどが装着されていれば，自動的にそれらのフロントプロセッサを起動します．

MSX-DOS2に搭載されている漢字ドライバと，MSX2+に登載されている漢字ドライバとは基本的に同じものです．また，各社からMSX-JE対応のフロントプロセッサと漢字ドライバをセットにしたカートリッジが発売されていますが，それらも同様です．

したがって，MSX2+にMSX-DOS2カートリッジを挿入すると，漢字ドライバが2つ同時に存在することになるのですが，その際には，MSX2+本体に内蔵されている漢字ドライバの方が起動します．MSX2+に内蔵されてい

る漢字ドライバと，MSX-DOS2の漢字ドライバの違いは，単漢字変換用の辞書の大きさの違いのみです．

漢字ドライバの起動と利用

漢字ドライバの起動は，BASICのコマンドラインから，

```
CALL  KANJIn      (n=0～3)
```

と入力することによって行われます．nを指定することによって，漢字の画面モードを選択することができます．漢字の画面モードについては次節のコマンドリファレンスを参照してください．

■ BASIC上での利用

BASICからの利用は，今までの画面への文字出力とほとんど同じです．たとえば，

```
LOCATE 10, 10
```

で表示する場所を決めて，

```
PRINT  "漢字ドライバの漢字モードでの漢字表示"
```

のように記述するだけで，漢字を出力できます．ですから，BASICで作ったプログラムならば，プログラム中のメッセージを漢字に変えるだけで，すぐに日本語対応のプログラムにすることができます．

日本語入力フロントプロセッサは，CTRL+SPACE(CTRLキーを押しながらSPACEキーを押す)またはGRAPH+SELECTでいつでも呼び出して使うことができます．

また，漢字を含んだ文字列の処理に必要なコマンドを拡張コマンドの形で用意しています．たとえば，全角文字と半角文字が混ざった文字列から，文字列を切り出す作業を，これまで使われてきたMID$で行うとすると，

```
MID$("全角文字ﾊﾝｶｸﾓｼﾞ", 2, 3)
```

となります．しかし，MID$は全角文字を1つの文字として認識することができません．全角文字は，漢字コード2バイトを1バイトごとに1つの文字であると判断してしまいます．その結果，この文の実行を実行すると，

```
PRINT MID$("全角文字ﾊﾝｶｸﾓｼﾞ", 2, 3)
S角
Ok
```

となってしまいます．この場合は，漢字ドライバの拡張コマンドであるCALL　KMIDコマンドを使います．このコマンドは全角文字も半角文字もどちらも1文字として判断しますから，

```
CALL KMID(A$,"全角文字ﾊﾝｶｸﾓｼﾞ", 2, 3) : PRINT A$
角文字
Ok
```

というように，正しく答えを出すことができます．

ほかにもLENやINSTRに相当するコマンドや，漢字コードを調べるコマンド等が用意されていますから，必要に応じて使うことができます．

■ MSX-DOS上での利用

MSX-DOS上でも漢字ドライバを使うことができます．ただし，漢字ドライバはBASIC内からしか起動することができないので，この場合は，MSX-DOSからいったんBASICにして，

```
CALL　KANJIn
```

を実行します．そして漢字モードにしたまま，

```
CALL　SYSTEM
```

でMSX-DOSに戻れば，MSX-DOS上で漢字ドライバを使うことができます．

フロントプロセッサはBASICと同様にCTRL＋SPACEまたはGRPH＋SELECTで呼び出すことができます．BIOSコールを使って入出力をしているプログラムならば，そのまま変更なく動くはずです．

漢字ドライバ拡張BASIC コマンドリファレンス

ここでは，漢字ドライバの組み込みにより拡張されるBASICのコマンドについて解説していきます．漢字ドライバでは，全角半角混在での文字列制御用の関数を，拡張ステートメントの形で提供しています．

拡張されるコマンドは次のとおりです．

CALL　KANJI
CALL　ANK
CALL　AKCNV
CALL　JIS
CALL　SJIS
CALL　KACNV
CALL　KEXT
CALL　KINSTR
CALL　KLEN
CALL　KMID
CALL　KNJ
CALL　KTYPE
CALL　CLS
CLS
CALL　PALETTE
WIDTH

では，順を追って解説していきましょう．

CALL KANJI

●書式● CALL KANJI [n]

漢字ドライバの起動と初期化をします. nは0から3までのうちどれか1文字で, 漢字の画面への表示方法を指定します. nを省略すると0とみなされます.

CALL KANJI0(またはCALL KANJI)では, 漢字フォントをMSXの標準漢字ROMから取り出し表示します. 文字サイズは縦16×横16ドットで表示します.

CALL KANJI1は, 漢字フォントをMSX標準漢字ROMから取り出し, 横サイズを12ドットに圧縮して表示します. 文字サイズは縦16×横12ドットです. ただし, 松下電器の仕様による12ドットフォントROMがスロットにセットされている場合, あるいはシステムに組み込まれている場合は, このROMからフォントを取り出して表示します.

CALL KANJI2はCALL KANJI0と文字サイズは同じですが, インターレースモードで表示するので, 縦方向の表示文字数が増えます.

CALL KANJI3はCALL KANJI1と文字サイズは同じですが, インターレースモードで表示するので, 縦方向の表示文字数が増えます.

日本語フロントエンドは, このステートメントの実行時またはシステムの立ち上げ時に組み込まれます.

MSXが起動してから一番最初のCALL KANJIの実行時には注意が必要です. その時点でのHIMEM(CLEAR文による)の設定はキャンセルされ, すべての変数およびソフトウェアスタック(FOR／NEXT, GOSUB／RETURN用)がクリアされます. これは漢字ドライバのワークエリアを(もしあれば日本語フロントエンドも)常駐させる処理が行われるからです. 2回目以降の実行は問題なく行われます.

漢字テキストモード

CALL KANJI命令などを実行し漢字表示ができる状態を〝漢字モード〟と呼びます. さらに漢字モードでかつSCREENステートメントで0もしく

は1が設定されている状態を〝漢字テキストモード〟と呼びます．BASICのコマンド待ちの状態では漢字テキストモードになっており，漢字の入出力が可能です．

CALL　KANJI実行後のWIDTHはANKモードでのWIDTHをもとに，以下のように決定されます．

・ANKモードでSCREEN 0を使用していた場合
　KANJI0かKANJI2のときはANKモード時のWIDTHの値の5分の4
　KANJI1かKANJI3のときはANKモード時のWIDTHの値

・ANKモードでSCREEN1を使用していた場合
　KANJI0かKANJI2のときはANKモード時のWIDTHの値
　KANJI1かKANJI3のときはANKモード時のWIDTHの値の5分の4

ただし，計算によって決まったWIDTHが26未満のときは26に設定されます．

漢字グラフィックモード

漢字モードでかつSCREENステートメントで2から11が設定されている状態を〝漢字グラフィックモード〟と呼びます．漢字グラフィックモードではプログラムによる漢字の出力のみが可能です．

漢字グラフィックモードの画面表示は，つねに表示できる最大に設定されます．WIDTHによるSCREENモードの選択は以下のようになります．

・KANJI0かKANJI2で，WIDTHが26～32のときは256ドットモード(VDPのモードでいうとSCREEN5)が，33～64のときは512ドットモード(VDPのモードでいうとSCREEN7)が選択される．
・KANJI1かKANJI3で，WIDTHが26～40のときは256ドットモード(VDPのモードでいうとSCREEN5)が，41～80のときは512ドットモード(VDPのモードでいうとSCREEN7)が選択される．

VRAM64KのMSX2の場合にはSCREEN6が使用されます．

漢字入力

漢字ドライバは仮想端末入力インターフェイスを持つカートリッジが存在する場合は，起動時にこれをインストールします．直接入力モード(ANK)と間接入力モード(漢字)は，CTRL+SPACE もしくは GRAPH+SELECT よって切り替えられます．仮想端末入力インターフェースが存在しないときは，単漢字変換機能が使用できます．

リスト 3.1 漢字テキストモードと漢字グラフィックモード

```
100 CALL KANJI3
110 CLS
120 LOCATE 0,10:PRINT "漢字テキストモード"
130 IF INKEY$="" THEN 130
140 SCREEN 7,,,,,3:SET PAGE 1,0
150 CLS
160 LOCATE 0,10:PRINT "漢字グラフィックモード"
170 IF INKEY$="" THEN 170
180 END
```

CALL ANK

●書式● CALL ANK

漢字ドライバを終了します．漢字ドライバを起動したときの SCREEN モードに戻ります．ただし，漢字ドライバ用に確保されたワークエリアなどのメモリは解放されません．

リスト 3.2 漢字ドライバ終了後のフリーエリアの表示

```
100 PRINT FRE(0)
110 IF INKEY$="" THEN 110
120 CALL KANJI3
140 PRINT FRE(0)
150 IF INKEY$="" THEN 150
160 CALL ANK
170 PRINT FRE(0)
180 END
```

CALL　AKCNV

●書式●　CALL　AKCNV(〈文字変数〉,〈文字列〉)

〈文字列〉中のすべての文字を全角文字に変換して〈文字変数〉に代入します.

CALL　JIS

●書式●　CALL　JIS(〈文字変数〉,〈文字列〉)

〈文字列〉の最初の2バイトを16進4桁のJISコードに変換して〈文字変数〉に代入します.

リスト3.3　文字列をPUTKANJIで表示する

```
100 SCREEN 7:CLS
110 A$="今日は良い天気というテスト"
120 CALL KLEN(L,A$)
130 X=100
140 FOR I=1 TO L
150   CALL KMID(K$,A$,I,1)
160   CALL JIS(J$,K$)
170   PUT KANJI (X,100),VAL("&H"+J$),15
180   X=X+16
190 NEXT I
200 IF INKEY$="" THEN 200
210 END
```

CALL SJIS

●書式● CALL SJIS(〈文字変数〉, 〈文字列〉)

〈文字列〉の最初の2バイトを16進4桁のシフトJISコードに変換して〈文字変数〉に代入します.

CALL KACNV

●書式● CALL KACNV(〈文字変数〉, 〈文字列〉)

〈文字列〉中の半角文字に変換できるすべての文字を半角文字に変換して〈文字変数〉に代入します.

CALL KEXT

●書式● CALL KEXT(〈文字変数〉, 〈文字列〉, 〈機能〉)

〈機能〉が0なら〈文字列〉中の半角文字だけを, 1なら全角文字だけを抜き出し〈文字変数〉に代入します.

CALL KINSTR

●書式● CALL KINSTR(〈数値変数〉[, 〈数式〉], 〈文字列1〉, 〈文字列2〉)

文字列の中から指定する文字列を捜してその文字の位置を返します.
BASICのINSTRステートメントと同じ動作をしますが, 全角文字(2バイト文字)に対応しています. 全角文字と半角文字が混ざっていても, 文字数単位の結果が正しく得られます.

〈文字列1〉の中から〈文字列2〉を捜しだし，見つかれば発見した位置を〈数値変数〉に代入します．見つからなければ0を〈数値変数〉に代入します．

〈数式〉は捜し始める位置を文字数を単位として指定します．0から255の範囲で指定します．省略されると1とみなし〈文字列1〉の左端から捜し始めます．

〈文字列2〉の長さが〈文字列1〉より大きかったり，〈文字列1〉がヌルストリングであれば0を〈数値変数〉に代入します．

リスト3.4　空白を区切り文字として捜し出し文字列を分割

```
100 SCREEN 0:CALL KANJI0
110 A$="文字の 区切りは スペースよ という テスト ":PRINT A$
120 CALL KLEN(L,A$,0)
130 J=1
140 CALL KINSTR(I,J,A$," ")
150 CALL KMID(K$,A$,J,I-J)
160 PRINT K$
170 J=I+1
180 IF J<L THEN 140
190 END
```

CALL　KLEN

●書式●　CALL　KLEN(〈数値変数〉，〈文字列〉[，〈機能〉])

文字列の総文字数を返します．BASICのLENステートメントと同じ動作をしますが，全角文字に対応しています．全角文字と半角文字が混ざっていても，正しい結果が得られます．〈機能〉を指定することによって，対象となる文字の種類を指定することができます．

〈機能〉が0または省略された場合〈文字列〉の全体の長さを〈数値変数〉に代入します．

〈機能〉が1のときは〈文字列〉中のすべての半角文字の文字数の合計を〈数値変数〉に代入します．

〈機能〉が2のときは〈文字列〉中のすべての全角文字の文字数の合計を〈数値変数〉に代入します．

リスト3.5　入力した文字列の文字数を表示させる

```
100 INPUT "文字列を入力してください";A$
110 CALL KLEN(A,A$,0)
120 CALL KLEN(B,A$,1)
130 CALL KLEN(C,A$,2)
140 PRINT A$;"は";A;"文字です"
150 PRINT "半角文字は全部で";B;"文字です"
160 PRINT "全角文字は全部で";C;"文字です"
170 END
```

CALL KMID

●書式●　CALL　KMID(〈文字変数〉，〈文字列〉，〈数式1〉[，〈数式2〉])

文字列の中から指定した長さの文字列を取り出して変数に代入します．

〈文字列〉の中の〈数式1〉番目の文字から〈数式2〉文字分だけ抜き出して〈文字変数〉に代入します．〈数式2〉が省略された場合は〈数式1〉番目の文字から終わりまでのすべての文字が代入されます．

リスト3.6　横書きを縦書きに変換する

```
100 DIM A$(10),B$(5)
110 FOR I=0 TO 9
120 READ A$(I)
130 NEXT I
140 FOR I=0 TO 9
150   PRINT A$(I)
160 NEXT I
170 PRINT
180 FOR I=1 TO 5
190   FOR J=9 TO 0 STEP -1
200     CALL KMID(C$,A$(J),I,1)
```

```
210     B$(I)=B$(I)+C$
220   NEXT J
230 NEXT I
240 FOR I=1 TO 5
250   PRINT B$(I)
260 NEXT I
270 END
280 DATA あいうえお
290 DATA かきくけこ
300 DATA さしすせそ
310 DATA たちつてと
320 DATA なにぬねの
330 DATA はひふへほ
340 DATA まみむめも
350 DATA やゐゆゑよ
360 DATA らりるれろ
370 DATA わ　を　ん
```

CALL KNJ

●書式● CALL KNJ(<文字変数>, <文字列>)

<文字列> で指定される4桁の漢字コードに相当する漢字1文字を <文字変数>に代入します．漢字コードが8000H未満のときは，JIS，それ以上のときはシフトJISとみなします．

リスト3.7 JIS第1水準の漢字コード表を作る

```
100 '第一水準漢字を表示する
130 K=&H21
135 PRINT
190 PRINT "      0 1 2 3 4 5 6 7 8 9 A B C D E F"
220   FOR KL=2 TO 7
240     PRINT HEX$(K*256+KL*16);"  ";
270     FOR X=0 TO 15
310       CALL KNJ(K$,HEX$(K*256+KL*16+X))
315       PRINT K$;
320     NEXT X
```

```
325     PRINT
330   NEXT KL
350 IF INKEY$="" THEN 350
360 K=K+1:IF K<>&H50 THEN 135
370 END
```

CALL KTYPE

●書式● CALL KTYPE(〈数値変数〉,〈文字列〉,〈数式〉)

〈文字列〉の中の〈数式〉番目の文字のタイプ(半角なら0, 全角なら1)を〈数値変数〉に代入します.

リスト3.8 文字列を半角と全角で色分けして表示

```
100 A$="究極ﾉ漢字処理ﾊ色ﾀﾞﾄｲｳﾃｽﾄ"
110 CALL KLEN(L,A$,0)
120 FOR I=1 TO L
130 CALL KTYPE(A,A$,I)
140 IF A=1 THEN COLOR 8 ELSE COLOR 7
150 CALL KMID(K$,A$,I,1)
160 PRINT K$;
170 NEXT I
180 END
```

CALL CLS

●書式● CALL CLS

画面をクリアーします. CALL KANJI ステートメントにより漢字テキストモードにしたときに, 画面をクリアするために使用します. 漢字モードでないときも使用できます.

CLS

●書式●　CLS

画面をクリアします．漢字テキストモードで，CLS ステートメントを実行すると，"Illegal function call"となり使用できません．漢字テキストモードのときは，CALL CLS ステートメントを使用してください．

なお，漢字グラフィックスモードでは CLS, CALL CLS ともに使用できます．

CALL　PALETTE

●書式●　CALL　PALETTE [(〈パレット番号〉，〈赤輝度〉，〈緑輝度〉，〈青輝度〉)]

カラーパレットの初期化または設定を行います．

漢字テキストモードにしたときに，パレットを変更するために使用します．漢字モードでないときでも使用できます．

パラメータをすべて省略した場合は，パレットを初期状態にします．

パラメータが指定された場合，〈パレット番号〉で指定されたパレットを〈赤輝度〉，〈緑輝度〉，〈青輝度〉の色に設定します．

COLOR ステートメント機能のうち，パレットを設定する機能は CALL KANJI ステートメントにより画面を漢字テキストモードにすると"Illegal function call"となり使用できません．漢字テキストモードのときは，CALL PALETTE ステートメントを使用してください．漢字グラフィックモードではエラーとならず使用できます．

WIDTH

●書式● WIDTH 〈桁数〉

1行に表示する文字数を設定します.

WIDTHステートメントは漢字モードのときには以下のように動作します.

・漢字テキストモードの場合

KANJI0かKANJI2のとき,〈桁数〉の指定は26から64が有効

KANJI1かKANJI3のとき,〈桁数〉の指定は26から80が有効

上記以外の値が指定されると"Illegal function call"となる

・漢字グラフィックモードの場合

つねに"Illegal function call"となる

3 日本語処理機能パワフル活用

漢字モード時に使い方が変わるBASICコマンド

漢字グラフィックモードではPRINT文によってグラフィック画面への文字出力が可能になりました．ただしこれを使用する際には以下の注意が必要です．

LOCATE文で指定するカーソルポジションは，半角文字単位です．漢字は，偶数位置にも奇数位置にも正しく表示できますが，行末では，半角2つ分スペースがないと表示できません．その場合は1行改行して行の先頭に表示していきます．

たとえば，KANJI1またはKANJI3の場合はX＝79，KANJI0またはKANJI2の場合はX＝64のポジションで漢字を表示しようとすると，そこには半角1文字分のスペースしかないので，1行改行して次の行の先頭に漢字を描きます．

カーソルポジションが表示できる最下行の一番右側になった場合，1行スクロールアップして，最下行の先頭から表示していきます．

カーソルポジションを保持しているエリアは1つしかないので，インターレースモードなどでアクティブページを変更しながらPRINTしていくと，他のページでのカーソルポジションの設定の影響を受けます．

インターレースモード(KANJI2，KANJI3)を設定している場合，PRINT文の文字出力は，インターレースモード用に漢字フォントの奇数ラインと偶数ラインを分離して，2つのページに書き込んでいます．文字出力をする際は，SCREEN文でEVEN／ODDのインターレースモード(SCREEN,,,,,3)に設定して，SET　PAGE文で表示ページを奇数ページに，アクティブ

ページを表示ページ-1に設定しなければなりません．
たとえば漢字グラフィックモードのSCREEN7を使うのであれば，

```
SCREEN7,,,,,3:SET PAGE1,0
```

をかならず入れてください．なお，SCREEN2から4では1ページしかないのでこの設定は意味を持ちません．

従来の方法であった，OPEN文を用いてPRINT #で表示する方法も同時に使用することは可能ですが，ANK文字しか表示できません．漢字を表示しようとした場合，グラフィック文字が表示されてしまいます．

PRINT文で表示するときのカラー指定は，直前のCOLOR文による前景色，背景色で行います．COLOR文で設定された前景色で文字を表示し，背景色で文字の背景を表示します．SCREEN2からSCREEN12までこの方法で表示します．

従来SCREENモードを変更しただけでは，スプライト表示禁止ビット(VDPレジスタ8)やインターレースモード(VDPレジスタ9)は変化しませんでしたが，漢字モードでは両方ともクリアされてしまいます．つまり，SCREENモードを変更すると，スプライトの表示は禁止され，インターレースモードはノンインターレースモードになるということです．ただしインターレースモードに関しては，SCREEN文の第6パラメータを同時に指定した場合は，正しく設定されます．

単漢字変換機能

漢字ドライバには標準で単漢字変換機能が組み込まれています．これは，仮想端末入力インターフェイス(MSX-Writeのようなフロントプロセッサ．MSX-JEのこと)がないシステムであってもBASICおよびMSX-DOS上で漢字の入力を行うことができるようにするための機能で，これを用いることによって，プログラムやテキスト等で全角文字を取り扱うことができます．ただしシステム上に仮想端末入力インターフェイスがある場合は，そちらが優先的に動作します．現在発売されているMSX2+の中にはすでにMSX-JEが内蔵されている機種もありますので，そういう機種をお持ちの方はこの機能を使うことはないでしょう．

■ 単漢字変換フロントプロセッサの特徴

単漢字変換とは漢字の読みによって，その読みを持つ漢字をすべて表示し，その中から漢字を選ぶ方式で，漢字変換の方法としては一番基本的なものの部類に当たります．しかし，どんなに高級な連文節変換やAI変換であっても，辞書にその文字が登録されていなかった場合は，単漢字変換で文字を入力していかなければなりません．文章を入力するのはちょっと辛いのですが，プログラムのなかに漢字のコメントやメッセージを入れるぐらいの使用法であれば充分なものといえるでしょう．

この単漢字変換フロントプロセッサは以下の特徴を持っています．

- 変換は漢字の音読みによって行う
- カーソルキーを使って，容易に目的の漢字や特殊文字を選択することができる
- JIS第2水準漢字に対応している

■ 単漢字入力モード

CALL　KANJIを実行し，漢字モードでCTRL+SPACEキー(CTRLキーを押しながらSPACEキーを押す)またはGRAPH+SELECTキーを

押すと，画面の最下行が反転して漢字変換ウィンドウが表示されます．この状態が単漢字入力モードで，漢字の入力が可能になります．

この状態で，CTRL＋SPACE（あるいはGRAPH＋SELECT）をもう一度押すと単漢字入力モードを終了することができます．画面最下行にファンクションキーの内容を表示するように設定していた場合はそれが表示されます．

もし画面の最下行で文字入力中だった場合（ファンクションキーの内容表示をOFFに設定していた場合など）は1行スクロールアップしてから最下行を反転し漢字入力ウィンドウを表示します．

■ 単漢字の入力方法

前記の方法で単漢字入力モードに移行すると，最下行にかな漢字変換ウィンドウが反転表示され，単漢字変換が可能になります．

ここでは「漢字」という文字を入力する場合を例にあげて，実際にかな漢字変換で漢字を入力する方法を説明して行きます．

①CTRL＋SPACEによって漢字入力モードにします．漢字入力モードにはいると画面最下行が反転し，漢字変換ウィンドウが表示されます．

②かなキー，もしくは　SHIFT＋かなキー（ローマ字入力モード）を押して，「か」を入力します．

すると変換ウィンドウに，

か　化下家可価歌加科火果花課河何過

のように，ひらがなの「か」および「か」と読む漢字が表示されます（表示される文字数は最大15文字．文字数は漢字モード，WIDTHの設定値によって異なる）．

③続いて「ん」を入力すると，

かん　間関完館官観管感監環巻刊幹干歓

のように，「かん」および「かん」と読む漢字が表示されます．

④変換ウィンドウ内のカーソルをカーソルキーの上，下，左，右キーまたはスペースキーで移動し，目的の「漢」の字の上に合わせて，リター

ンキーを押して決定します．すると，画面に「漢」が表示され，変換ウィンドウ内はクリアされます．

⑤次に「し」，「〃」を入力します．（ローマ字変換の場合は，「J」，「I」を入力する）

⑥同様にカーソルを「字」に合わせてリターンキーを押す．

■ JISコード入力

前記の方法で単漢字入力モードに移行すると，最下行に仮名漢字変換ウィンドウが反転表示され，JIS コード入力が可能になります．JIS コードで入力する場合は，CTRL キーを押しながら 4 桁の JIS コードを連続して入力します．するとその JIS コードに相当する文字から最大 127 文字が入力の候補として選ばれ，JIS コードの順に変換ウィンドウに最大 15 文字ずつ表示されます．変換ウィンドウ内のカーソルをカーソルキー上，下，左，右またはスペースキーで移動し，目的の文字に合わせて，リターンキーで決定します．すると画面にその文字が表示され，変換ウィンドウ内はクリアされる．

ここで「☆」という記号を入力する場合を例にあげて，JIS コード入力の方法を説明します．

① CTRL＋SPACE によって漢字入力モードにはいります．画面最下行が反転表示され，変換ウィンドウが表示されます．

② CTRL キーを押しながら JIS コードを入力します．「☆」の場合 JIS コードは 2179 ですから，CTRL キーを押しながら 2179 と入力します．すると変換ウィンドウに，

☆★○●◎◇◆□■△▲▽▼※〒

と表示されます．

③変換ウィンドウ内のカーソルをカーソルキー上，下，左，右キーまたはスペースキーで移動し，「☆」の上に合わせて，リターンキーを押して決定します．すると画面にその文字が表示され，変換ウィンドウ内はクリアされます．

■ 変換中の訂正

変換中の訂正はESCキーを押すことで可能です．ESCキーを押すと変換ウィンドウ内がクリアされ，読み入力の状態に戻ります．

■ 各種文字の入力方法

その他各種文字の入力は以下のようになっています．

- 全角ひらがな　「かな」ランプが点灯している状態で入力
- 全角カタカナ　単漢字入力モードで「かな」,「CAPS」(CAPS　LOCK) キーの両方が点灯している状態で入力
- 半角カタカナ　単漢字入力モードを解除して,「かな」,「CAPS」(CAPS LOCK) キーの両方が点灯している状態で入力
- 全角記号　JISコード入力を使用して入力
- 全角英数字　単漢字入力モードで「かな」ランプが点灯していない状態で入力

MSX2＋ではJIS第2水準漢字ROMはオプションですが，システムにJIS第2水準漢字ROMが存在する場合は，自動的にJIS第2水準対応の辞書が選択されますので，第2水準の漢字も入力可能です．

ローマ字変換機能については，MSX2本体のローマ字変換機能と同様です．SHIFT＋かなキー(SHIFTキーを押しながらかなキーを押す)によってローマ字入力が可能になります．

活用サンプルプログラム①
漢字スケジュール・メモ

ここでは，漢字テキストモード上で動作するサンプルプログラムを2つ紹介します．紹介するプログラムはBASIC上で動作する簡単なものですから，市販のソフトなどとは実用性や速度などの面では比べものになりません．とはいえ，気軽に使うことができ，簡単に改良できるという自作プログラムならではの楽しみ方もできるでしょう．

最初に紹介するプログラムは，日々のスケジュールを書き込んでおく，メモ・プログラムです．実用性といった面では疑問があるかもしれませんが，手帳の感覚でなんでも書き込んでおいて，内容をディスクに保存したり，プリンタに出力したりできるようになっています．

■ スケジュール・メモの使い方

プログラムをリスト3.9に示します．このプログラムを〝RUN〟させると，まずMSXの時計に設定されている日付を表示し，今日の日付を入力するよう促します．日付が合っていればそのままリターンキーを押せばよいのですが，変更したい場合は，西暦と月と日を2桁づつ(西暦は下2桁)を〝／〟(スラッシュ)で区切って入力し，リターンキーを押してください．

日付の入力が終わると，ディスクからスケジュールの保存されているファイル〝schdul.dat〟を読み込んで，今日の予定を表示しコマンド待ち状態となります．最初に立ち上げたときには，スケジュールのはいったファイルは存在しませんから，枠だけを表示してコマンド待ち状態となります．この状態で受け付けるコマンドは次の表のようになっています(表3.2)．

ここで，1つ注意することがあります．表3.2にあるように，コマンドはファンクションキーによっても実行できます．ただし，これらを使うことができるのは，フロントエンドプロセッサを切ってあるときだけです．これは，フロントエンドプロセッサが，変換や確定などの目的でファンクションキーを利用しているためです．ですから，漢字を入力したあとは英字のコマンドを使うか，CTRL＋スペースキーやGRAPH＋SELECTキーによってフロントエンドを切ってからファンクションキーを使うようにしてください．

コマンド		機　能
カーソル移動	→ ←	ページ切り換え
	↑ ↓	カーソルの移動
入力・編集など	F1 W	データの入力と修正
	F2 S	日付による検索
	F3 I	カーソル位置にデータを挿入
	F4 D	カーソル位置のデータを削除
	F5 P	スケジュールを印刷
	F10 E	ファイルの出力と終了

表 3.3　スケジュール・メモのコマンド一覧

では，それぞれのコマンドの説明をしましょう．データを書き込むには，カーソル(バックが緑色に表示されている行)をカーソル・キーの上下を使って適当な場所に動かして，W キー(入力)を押します．すると上段に質問がでますから，日付，時刻，メモの順に入力すると，カーソル位置にデータが書き込まれます．データは横1行を1レコードとして，そのレコードを3つにわけて管理されています．日付と時刻はそれぞれ半角で5文字，メモ欄は半角48文字(全角ならば24文字)という表示文字数からくる制限があります．

挿入のIキーや，削除のDキーも入力の場合と同様カーソルのある行に対して行われます．画面に表示される行は11行ですが，書き込める行は最大50行と設定してあり，カーソルキーで画面内の移動が出来るだけでなく，前後にスクロールさせることもできます．また，カーソルキーの左右はページ単位の切り換えに使います．

照会のSキーは指定する日付の検索のためのもので，先頭から検索して入力した日付と一致する行があるかどうかを探します．見つかった場合，その行以降を画面に表示します．印刷のPキーは，カーソル位置から1画面分(11行)のメモをプリンタに出力します．入力や修正，削除などの作業のあとは，必ずEキーを押してプログラムの実行を終了するようにしてください．これは，最初にデータを1度に読み込んでおき，終了時に書き込んで更新するようにしているからです．

■ プログラムの流れ

プログラムの基本的な流れは次のようになっています．まず，日付の入力を行い，初期画面を書きます．既存のデータファイルがある場合は，それを読み込み，その日の予定を表示します．その後は，メインルーチンでコマンド入力を判断して，それぞれのサブルーチンへと分岐する繰り返しです．終了時には，データを再びファイルに書き戻して，ファンクションキーの初期化(BIOSの&H3Eを利用)などを行って終了します．

■ 漢字入力ためのテクニック

漢字ドライバを利用したプログラムでは，漢字の入力と出力を漢字ドライバに任せるのですから，漢字コードを正しく受け取り，きちんと処理さえすればよいことになります．その処理をこのプログラムで見てみることにしましょう．

1120行を見てください．ここでは，IN$に入力された文字列から日付を設定していますが，IN$が全角文字によって入力された場合を想定して，半角文字列に変換しているのです．これと同様の処理は，データ照会ルーチンでの日付の読み込み(2030行)，データの入力ルーチンでの日付や時刻の読み込み(2330行，2360行)で行っています．

また，メインループ中の1380行からのコマンド受け取り部分では，入力文字列がいったん途切れるまで取り込み，その後で先頭の1文字分を取り出すという方法をとっています．それは，ファンクションキーに設定されている文字列が長いために，その先頭を取り出すということと，全角文字を受け取った場合の半角への変換という2つの理由があるのです．

また，データ入力ルーチンの中で，メモ欄の文字列を48バイト以下にする処理(2390～2400行)も単純にLEFT$で取り出すのではないことに注意してください．

■ 画面表示のテクニック

KANJIテキストモードにおいても，ANKモードのときと同様のエスケープシーケンスがサポートされています(表3.4)．

○カーソル移動	
<ESC>A	カーソルを上に移動
<ESC>B	カーソルを下に移動
<ESC>C	カーソルを右に移動
<ESC>D	カーソルを左に移動
<ESC>H	カーソルをホームポジションに移動
<ESC>Y<Y座標+20H><X座標+20H>	カーソルを(X,Y)の位置に移動
○編集，削除	
<ESC>j	画面をクリア
<ESC>E	画面をクリア
<ESC>K	行の終わりまで削除
<ESC>J	画面の終わりまで削除
<ESC>L	1行挿入
<ESC>M	1行削除
○その他	
<ESC>x4	カーソルの形を'■'にする
<ESC>x5	カーソルを消す
<ESC>y4	カーソルの形を'_'にする
<ESC>y5	カーソルを表示する

表3.4　エスケープシーケンス表

本プログラムでは，このエスケープシーケンスのうち行末までの消去，行削除による上スクロール，行挿入による下スクロールを利用しています．

また，KANJIテキストモードでは，直前にCOLOR文で指定した色(前景色，背景色)によって文字が描かれるようになっています．ところが，KANJIテキストモードでCOLOR文を使うと，周辺色は背景色と同一の色として設定されてしまいます．このために，周辺色と異なった背景色を使おうとすると，そのたびに周辺色が変化してしまい，目障りなものとなります．そこで背景色だけを変えて周辺色を変えないようにするために，背景色のカラーコードが格納されているワークエリアに直接カラーコードを書き込むようにしています．この背景色のカラーコードが格納されているワークエリアはBAKCLR(&HF3EA)です．

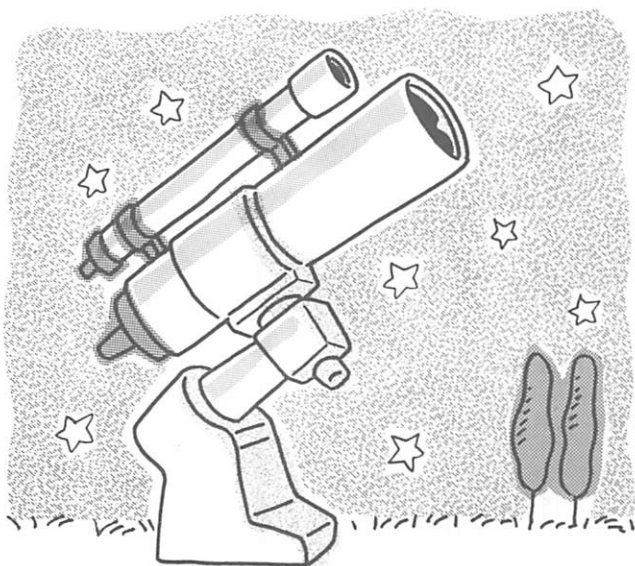
リスト 3.9　漢字スケジュール・メモ "SCHDUL.BAS"

```
1000 '   スケジュール管理プログラム
1010 '
1020 CALL KANJI0: WIDTH 64
1030 KEY ON: COLOR 15,1:CALL CLS
1040 MAXFILES=1: CLEAR 5000: DEFINT A-Z
1050 BC=&HF3EA: BK=1
1060 MX=50: DIM DT$(MX),TM$(MX),TX$(MX)
1070 ES$=CHR$(27): EC$=ES$+"K": EU$=ES$+"M": ED$=ES$+"L"
1080 '--- Input Date ---
1090 GET DATE D$: IN$=""
1100 LOCATE 10,2: PRINT "今日の日付は [";D$;"]です"
1110 LOCATE  6,4: INPUT "日付を入力してください(YY/MM/DD) ";IN$
1120 IF IN$<>"" THEN CALL KACNV(D$,IN$): SET DATE D$
1130 '--- Initialize Screen ---
1140 CALL CLS
1150 GOSUB 2630 'DispTopLine
1160 KEY 1,"W：入力": KEY 2,"S：照会": KEY  3,"I：挿入"
1170 KEY 4,"D：削除": KEY 5,"P：印刷": KEY 10,"E：終了"
1180 FOR I=6 TO 9: KEY I,"": NEXT I
1190 '--- Read File ---
1200 ON ERROR GOTO 1310 'NoData
1210 OPEN "schdul.dat" FOR INPUT AS #1
1220 FOR I=0 TO MX
1230   INPUT #1,DT$(I),TM$(I),TX$(I)
1240 NEXT I
1250 '--- End of File ---
1260 CLOSE #1
1270 ON ERROR GOTO 0
1280 SCX=0: DTX=0: CUR=1: GOSUB 1820 'DispPage
1290 GOTO 1330 'Main
1300 '--- No Data ---
1310   RESUME 1260
1320 '--- Main ---
1330 ON STOP GOSUB 2710: STOP ON
1340 K$=RIGHT$(D$,5): GOSUB 2040 'Search
1350 '
1360 COLOR 15,1: LOCATE 50, 0
1370 PRINT USING "ｶﾗﾑNo. : ###"; SCX+CUR;
1380 I$=INKEY$: IF I$="" THEN 1380
1390 J$=INKEY$: IF J$<>"" THEN I$=I$+J$: GOTO 1390
1400 CALL KMID(I$,I$,1,1): CALL KACNV(I$,I$)
```

```
1410 IF I$>="a" AND I$<="z" THEN I$=CHR$(ASC(I$)-&H20)
1420 IF I$=CHR$(30) THEN 1540
1430 IF I$=CHR$(31) THEN 1620
1440 IF I$=CHR$(28) THEN 1760
1450 IF I$=CHR$(29) THEN 1790
1460 IF I$="S" THEN 1990 'SHOW
1470 IF I$="W" THEN 2160 'WRITE
1480 IF I$="D" THEN 2450 'DELETE
1490 IF I$="P" THEN 2530 'PRINT
1500 IF I$="I" OR I$="ｲ" THEN 2210 'INSERT
1510 IF I$="E" OR I$="ｴ" THEN 2710 'END
1520 GOTO 1360
1530 '--- Up Cursor ---
1540 GOSUB 1730 'Cursor Off
1550 IF CUR>1 THEN CUR=CUR-1: GOTO 1590
1560   IF SCX=0 THEN BEEP: GOTO 1590
1570     SCX=SCX-1: LOCATE 0,1: PRINT ED$;
1580     I=11: GOSUB 1900 'DISP A Data
1590 GOSUB 1700 'Cursor On
1600 GOTO 1360  'Main
1610 '--- Down Cursor ---
1620 GOSUB 1730 'Cursor Off
1630 IF CUR<11 THEN CUR=CUR+1: GOTO 1670
1640   IF SCX=MX-11 THEN BEEP: GOTO 1670
1650     SCX=SCX+1: LOCATE 0,1: PRINT EU$;
1660     I=11: GOSUB 1900 'Disp 1 Data
1670 GOSUB 1700 'Cursor On
1680 GOTO 1360  'Main
1690 '--- Cursor On ---
1700 I=CUR: BK=12: GOSUB 1900 'DISP A Data
1710 RETURN
1720 '--- Cursor Off ---
1730 I=CUR: GOSUB 1900 'DISP A Data
1740 RETURN
1750 '--- Up Page ---
1760 SCX=SCX+10: GOSUB 1820 'DispPage
1770 GOTO 1360 'Main
1780 '--- Down Page ---
1790 SCX=SCX-10: GOSUB 1820 'DispPage
1800 GOTO 1360 'Main
1810 '--- Display Page ---
1820 IF SCX<0 THEN BEEP: SCX=0
1830 IF SCX>MX-11 THEN BEEP: SCX=MX-11
```

```
1840 FOR I=1 TO 11
1850   GOSUB 1900 'Displ
1860 NEXT I
1870 GOSUB 1700 'Cursor On
1880 RETURN
1890 '--- Display 1 Datum ( <I>th ) ---
1900 J=SCX+I-1: LOCATE 0,I
1910 COLOR 15,1: POKE BC,BK
1920 PRINT RIGHT$(SPACE$(5)+DT$(J),5);"  ";
1930 PRINT RIGHT$(SPACE$(5)+TM$(J),5);"  ";
1940 PRINT TX$(J); EC$;
1950 COLOR 9,1: POKE BC,BK
1960 LOCATE 6,I: PRINT "|";: LOCATE 13,I:PRINT "|";
1970 BK=1
1980 RETURN
1990 '--- Search Data ---
2000 COLOR 1,1: POKE BC,3: K$=""
2010 LOCATE  0, 0: PRINT "日付(xx/xx)";EC$;
2020 LOCATE 11, 0: INPUT K$: CALL KACNV(K$,LEFT$(K$,5))
2030 CALL KACNV(K$,K$): K$=LEFT$(K$,5)
2040 '
2050 I=0
2060 IF I>MX THEN 2090
2070   IF DT$(I)=K$ THEN 2110
2080 I=I+1: GOTO 2060
2090 BEEP: LOCATE 0,0: PRINT K$; "の予定はありません"; EC$;
2100 FOR W=1 TO 500: NEXT W: GOTO 2130
2110 SCX=I: CUR=1: GOSUB 1820 'DispPage
2120 '
2130 GOSUB 2630 'Disp Top
2140 GOTO 1360  'Main
2150 '--- Write Data ---
2160 DTX=SCX+CUR-1: GOSUB 2300 'GetData
2170 GOSUB 1820 'DispPage
2180 GOSUB 1620 'DownCursor
2190 GOTO 1360  'Main
2200 '--- Insert Data ---
2210 DTX=SCX+CUR-1
2220 FOR I=MX-1 TO DTX STEP -1
2230   DT$(I+1)=DT$(I): TM$(I+1)=TM$(I): TX$(I+1)=TX$(I)
2240 NEXT I
2250 GOSUB 2300 'GetData
2260 GOSUB 1820 'DispPage
```

```
2270 GOSUB 1620 'DownCursor
2280 GOTO 1360  'Main
2290 '--- Get Data ---
2300 COLOR 1,1: POKE BC,3
2310 LOCATE 13, 0: PRINT DT$(DTX); EC$; :K$=""
2320 LOCATE  0, 0: INPUT "日付(xx/xx)";K$
2330 CALL KACNV(K$,K$): DT$(DTX)=LEFT$(K$,5)
2340 LOCATE 13, 0: PRINT TM$(DTX); EC$; :K$=""
2350 LOCATE  0, 0: INPUT "時刻(xx:xx)";K$
2360 CALL KACNV(K$,K$): TM$(DTX)=LEFT$(K$,5)
2370 LOCATE  7, 0: PRINT TX$(DTX); EC$; :K$=""
2380 LOCATE  0, 0: INPUT "メモ ";K$
2390 IF LEN(K$)<=48 THEN 2410
2400   CALL KLEN(L,K$): CALL KMID(K$,K$,1,L-1): GOTO 2390
2410 TX$(DTX)=K$
2420 GOSUB 2630 'DispTopLine
2430 RETURN
2440 '--- Delete Data ---
2450 DTX=SCX+CUR-1
2460 FOR I=DTX TO MX-1
2470   DT$(I)=DT$(I+1): TM$(I)=TM$(I+1): TX$(I)=TX$(I+1)
2480 NEXT I
2490 DT$(MX)="": TM$(MX)="": TX$(MX)=""
2500 GOSUB 1820 'DispPage
2510 GOTO 1360  'Main
2520 '--- Print Out Page ---
2530 LPRINT TAB(9) ; "|" ; TAB(19) ; "|"; CHR$(13);
2540 LPRINT "  日付     時間                    メ     モ"
2550 PM=SCX+CUR+10:IF PM>MX THEN PM=MX
2560 FOR I= SCX+CUR-1 TO PM
2570   LPRINT "  "  ;      DT$(I);
2580   LPRINT TAB( 9); "| "; TM$(I);
2590   LPRINT TAB(19); "|" ; TX$(I)
2600 NEXT I
2610 GOTO 1360 'Main
2620 '--- Display Top Line ---
2630 COLOR 1,1: POKE BC,7
2640 LOCATE  0, 0: PRINT " 日付"; EC$;
2650 LOCATE  8, 0: PRINT "時刻";
2660 LOCATE 15, 0: PRINT "メ     モ"
2670 COLOR 9,1: POKE BC,7
2680 LOCATE 6,0: PRINT "|";: LOCATE 13,0:PRINT "|";
2690 COLOR 15,1: RETURN
```

```
2700 '--- FIN ---
2710 FOR I=1 TO 10: KEY(I) OFF: NEXT I
2720 STOP OFF
2730 OPEN "schdul.dat" FOR OUTPUT AS #1
2740 FOR I=0 TO MX
2750   PRINT #1,DT$(I); ","; TM$(I); ","; TX$(I)
2760 NEXT I
2770 CLOSE #1
2780 DEFUSR=&H3E : D=USR(0) 'Init func keys
2790 CALL CLS: END
```

活用サンプルプログラム②
簡易漢字エディタ

漢字ドライバを利用したサンプルプログラムのもう1つはテキストエディタです.テキストエディタは文字どおり文書を作成・編集するツールのことです.ワードプロセッサは,このテキストエディットの機能とプリンタに印刷するために整形して出力する機能を合わせ持ったものを指す場合が多いようです.

ここで紹介するプログラムは編集の機能の他に印刷の機能も合わせ持っています.ですから,ワープロと呼んでも差し支えないのかもしれませんが,印刷の機能が貧弱ということと,作者がテキストエディタという言葉が好きだという理由で,いちおう漢字エディタということにしておきます.

このプログラムでの目標は次の2つです.

- **インターレースモードを利用することで最大80文字×23行(半角文字)の表示を行う**
- **半角文字と全角文字を完全に取り扱う**

この目標を実現するためプログラムはかなり大きなものとなり,また処理の高速化のためにマシン語の助けを借りることとなりました.

■ プログラムの実行方法

このプログラムは,BASICで記述されたメインプログラムとマシン語のサブルーチン群の2つに分けられます.マシン語のプログラムは直接打ち込むことはできませんので,マシン語のバイナリファイルを作成するプログラムとエディタプログラム本体の2つが必要です.リスト3.10にバイナリファイル作成プログラムを,リスト3.11にエディタ本体のプログラムを示します.

リスト3.10を実行すると〝kedusr.bin〟というマシン語のバイナリファイルが作成されます.1度バイナリファイルが作成されるとこの作成のためのプログラムは必要なくなりますが,いざというときのために保存しておいた方がよいでしょう.

エディタを使うには，カレントディレクトリ上にこのファイルがあることを確認して，リスト 3.11 を実行すればよいのです．このプログラムは一部マシン語を利用していますので，プログラムミスによっては暴走の危険があります．ですから，自分で打ち込む場合は，実行する前に必ずプログラムをSAVEしておくべきです．また，このプログラムでは，テキスト領域をできるだけ多く確保するために，空き領域がほとんどありません．ですから余計なスペースや，コメントは入力しないようにしてください．

■ プログラムの操作

このプログラムには大きく分けて2つのモードがあります．実際に文字を打ち込んでいくエディットモードとファイル操作，印刷などを行うコマンドモードです．

コマンドモードは，ファイルの読み込み，書き込み，印刷など，文字入力以外の操作を行うモードです．起動時はこのモードになっています．使用できるコマンドとその機能の一覧を表 3.5 に示します．各コマンドはそれぞれのキーを押すことで実行されます．

エディットモードは実際の文字入力作業を行うモードです．基本的には，キーを押すとその文字が画面に反映されて，テキストが作成されていきます．もちろん漢字も入力することができます．このエディットモードで使用できる特殊なキーとその機能を次の表 3.6 にまとめておきます．

ここで注意する必要があるのは TAB キーとリターンキーの働きでしょう．これらは挿入モードと上書きモードでまったく異なっています．また，この挿入／上書きモードは BASIC 上のスクリーン・エディタと異なってカーソル移動や改行などでは変化しません．

■ 容量の制限

このエディタで扱うことのできるテキストの量には制限があります．具体的には，半角文字で約 8000 文字以内，そして行数としては 200 行以内というものです．なお，ここでの行数は実際の画面で表示されている行数のことでファイル中での改行の数ではありません．

コマンド	機　能
[E]	エディットモードへはいる
[P]	テキストをプリントアウトする
[S]	テキストをファイルに書き込む
[L]	ファイルを読み込む
[Q]	エディタの実行を終了する

表 3.5　コマンドモードで使用できるキーとその機能

キ　ー	機　能
[CTRL]+[E], [↑]	カーソルを上に移動
[CTRL]+[X], [↓]	カーソルを下に移動
[CTRL]+[D], [→]	カーソルを右に移動
[CTRL]+[S], [←]	カーソルを左に移動
[CTRL]+[W]	前ページに戻る
[CTRL]+[Z]	次ページに進む
[CTRL]+[R], [INS]	挿入／上書きモードの切り換え
[CTRL]+[M], [RET]	挿入モードでは行を2つに分割 上書きモードでは次行の先頭に移動
[CTRL]+[O]	行を2つに分割
[CTRL]+[I], [TAB]	挿入モードでは次のタブストップまで空白を挿入 上書きモードでは次のタブストップまで移動
[CTRL]+[G], [DEL]	カーソル位置の1文字を削除
[CTRL]+[H], [BS]	カーソルの左側の1文字を削除
[CTRL]+[Y]	カーソルのある行を削除
[CTRL]+[P]	削除された行を貼り付ける
[CTRL]+[L], [HOME]	画面を書き直す
[ESC]	コマンドモードへ戻る

表 3.6　エディットモードで使用できる特殊キーとその機能

これらの制限のため，巨大なテキストを編集するといった用途には使いづらいでしょうが，手紙やレポート，パソコン通信のアップロード用のファイル作成などパーソナルな用途に使う分には十分活用できるはずです．

■ プログラムの変更・解析

前にも述べましたように，このプログラムはメモリ領域を大量に使っているために，機能を拡張したりする余裕は残念ながらほとんどありません．しかし，漢字モードや横幅の設定(1020行)は自分の見やすいモードに変更しておくといいでしょう．たとえば，とにかくたくさん文字を表示させたいときは，

```
1020 CALL KANJI3： WIDTH 80
```

などとするとか，細かい字が読みづらい場合は，

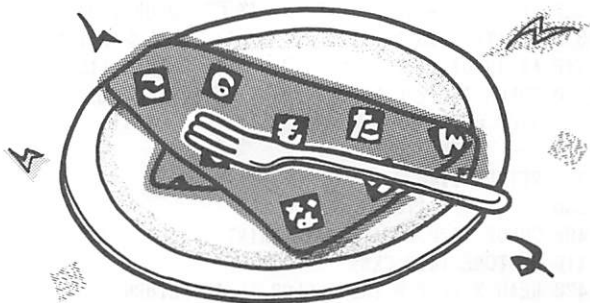
```
1020 CALL KANJI0： WIDTH 32
```

とするなどです．これ以外にも，色やパレットの変更，あるいはメッセージを自分用に差し換えることぐらいは簡単にできるでしょう．

また，マシン語サブルーチンの内容を知りたいという人のためにそれぞれのルーチンの内容を簡単に紹介しておきます．

- DEFUSR ＝&HCA02：S＝USR(A$)
  A$の1文字目が半角文字ならS=0，全角文字の1バイト目ならS=1を返す
- DEFUSR1＝&HCA04：POKE &HCA00，C：S＝USR1(A$)
  A$のC文字目が半角文字ならS=0，全角文字の1バイト目ならS=1，全角文字の2バイト目ならS=2を返す
- DEFUSR2＝&HCA06：L＝USR2(A$)：S＝PEEK(&HCA01)
  A$の先頭から何文字目までをテキストとして読み込めるかをLに返す．Sはその次の文字によって変化する(ファイルの読み込み用)

リスト3.10　バイナリファイル作成用プログラム〝MKUSR.BAS〞

```
100 ' Make "KEDUSR.BIN"
110 ' Written by TeK
120 clear 200,&hc9ff
130 st=&hca00: ad=st
140 read a$: if a$="*" then 200
150  poke ad,val("&H"+a$)
160  ad=ad+1
170 goto 140
200 bsave "kedusr.bin",st,ad-1
210 print "Complete!"
220 end
10000 DATA 01,00,18,04,18,13,18,57,FE,03,20,3D,EB,7E,B7,28
10010 DATA 34,23,5E,23,56,06,01,18,14,FE,03,20,2C,EB,7E,4F
10020 DATA 3A,00,CA,3D,B9,30,1E,3C,47,23,5E,23,56,1A,13,CD
10030 DATA 9A,CA,2E,01,38,05,2D,10,F4,18,10,10,02,18,0C,13
10040 DATA 10,EB,2C,18,06,2E,03,18,02,2E,04,26,00,3E,02,32
10050 DATA 63,F6,22,F8,F7,C9,3E,05,2E,00,32,01,CA,18,EC,FE
10060 DATA 03,20,F3,EB,7E,4F,23,5E,23,56,2E,00,79,B7,28,EA
10070 DATA 1A,FE,20,30,14,FE,0A,28,0C,FE,09,28,04,3E,02,18
10080 DATA D9,3E,03,18,D5,3E,04,18,D1,CD,9A,CA,30,07,0D,3E
10090 DATA 01,28,C7,13,2C,13,2C,0D,18,D2,FE,81,3F,D0,FE,FD
10100 DATA D0,FE,A0,D8,FE,E0,3F,C9
10110 data "*"
```

リスト 3.11　漢字テキストエディタ本体 "KED.BAS"

```
1000 ' MSX漢字ﾄﾞﾗｲﾊﾞ対応ﾃｷｽﾄｴﾃﾞｨﾀ
1010 ' Vers. 1.0     Written by TeK
1020 CALL KANJI2:WIDTH 64
1030 COLOR 15,12,12:CALL CLS:CALL PALETTE
1040 CALL PALETTE(4,1,1,3):CALL PALETTE(12,0,2,0)
1050 MAXFILES=1:CLEAR 8300,&HC9FF:DEFINT A-Z:POKE &HFAFD,&HC0
1070 DM=200:DIM L$(DM),LK(DM)
1080 PRINT "** 漢字ｴﾃﾞｨﾀ Vers. 1.0 **":PRINT:PRINT "-- 仕事中 --"
1100 BLOAD "kedusr.bin"
1110 AD=&HCA00:DEFUSR=AD+2:DEFUSR1=AD+4:DEFUSR2=AD+6
1120 GOSUB 1550:ON STOP GOSUB 1380:STOP ON
1130 '--- ﾒｲﾝ･ﾙｰﾌﾟ
1140 COLOR 15,12:FOR I=0 TO YM-1:GOSUB 3190:NEXT I
1150 GOSUB 1400
1160 A$=INKEY$:IF A$="" THEN 1160 ELSE CM=(INSTR(CK$,A$)+1)/2
1170 IF CM=0 THEN 1160 ELSE ON CM GOTO 1200,1220,1240,1260,1320
1190 '** Edit **
1200 I=YM:GOSUB 3190:GOSUB 3250:GOSUB 1620:GOTO 1150
1210 '** Print **
1220 C$="印刷":GOSUB 1500:GOSUB 3310:GOTO 1150
1230 '** Save **
1240 C$="保存":GOSUB 1500:GOSUB 3420:GOTO 1150
1250 '** Load **
1260 C$="読込":GOSUB 1500:IF SF THEN 1300
1270 LOCATE 8,YS:PRINT "ﾃｷｽﾄが保存されていません";
1280 LOCATE 8,YM:PRINT "本当ですか(Y/N)？";
1290 A$=INPUT$(1):IF INSTR("Yy",A$)=0 THEN 1150
1300 GOSUB 3530:GOTO 1140
1310 '** Quit **
1320 IF SF THEN 1350 ELSE C$="終了":GOSUB 1500
1330 LOCATE 2,YS:PRINT "保存しないで止めるの(Y/N)？";
1340 A$=INPUT$(1):IF INSTR("Yy",A$)=0 THEN 1150
1350 COLOR 15,12:CALL CLS:PRINT "お疲れ様 .またどうぞ.";
1360 STOP OFF:END
1370 '--- [CTRL]+[STOP]
1380 RETURN 1140
1390 '--- ｺﾏﾝﾄﾞ行の表示
1400 GOSUB 1520:LOCATE 0,YM:PRINT "コマンド？";
1410 RESTORE 1470:CK$=""
1420 READ X:IF X<0 THEN COLOR 15,12:RETURN
1430 READ Y,K$,C$:LOCATE X,Y+YM
```

```
1440 CK$=CK$+K$+CHR$(ASC(K$)+32):CALL AKCNV(K$,K$)
1450 COLOR 7,12:POKE BC,6 :PRINT K$;
1460 COLOR 1,12:POKE BC,11:PRINT C$;:GOTO 1420
1470 DATA 14,0,"E","編集",22,0,"P","印刷"
1480 DATA  6,1,"S","保存",14,1,"L","読込",22,1,"Q","終了",-1
1500 GOSUB 1520:COLOR 7,12:POKE BC,6
1510 LOCATE 0,YM:PRINT "["C$"]";:COLOR 1,12:POKE BC,11:RETURN
1520 LOCATE 0,YM,0:COLOR 1,12:POKE BC,11
1530 PRINT ES$"K ":PRINT ED$;:RETURN
1540 '--- 変数の初期化
1550 KY$="":RESTORE 1720
1560 READ D:IF D=-1 THEN 1570 ELSE KY$=KY$+CHR$(D):GOTO 1560
1570 XS=PEEK(&HF3B0):XM=XS-1:YS=PEEK(&HF3B1)-2:YM=YS-1:BC=&HF3EA
1580 ES$=CHR$(27):ED$=ES$+"K":DD$=CHR$(127):FL$="TMP"
1590 FOR I=0 TO DM:L$(I)="":LK(I)=0:NEXT I
1600 CX=0:CY=0:LY=0:EL=0:MF=-1:SF=-1:CB$="":CK=0:RETURN
1610 '--- 編集ﾓｰﾄﾞ
1620 LOCATE CX,CY:C$=INPUT$(1):LOCATE ,,0
1630 IF C$<" "OR C$=DD$ THEN 1680 ELSE PRINT C$;
1640 T$=INKEY$:IF T$="" THEN GOSUB 2550:GOTO 1620
1650 IF T$<" " OR T$=DD$ THEN GOSUB 2550:C$=T$:GOTO 1680
1660 PRINT T$;:C$=C$+T$:GOTO 1640
1670 '--- Control Code
1680 IF C$=DD$ THEN GOSUB 1790:GOTO 1620
1690 IF C$>ES$ THEN ON ASC(C$)-27 GOSUB 2080,2150,2190,2270:GOTO 1620
1700 ON INSTR(KY$,C$) GOSUB 2080,2150,2190,2270,1790,1760,2470,3170,28
30,2860,3140,1730,2230,2010,2350
1710 IF C$<>ES$ THEN 1620 ELSE RETURN
1720 DATA 4,19,5,24,7,8,9,12,13,15,16,18,23,25,26,-1
1730 '--- 挿入／上書切り換え
1740 MF=NOT MF:GOSUB 3250:RETURN
1750 '--- BS
1760 IF CX=0 AND LY+CY=0 THEN RETURN ELSE GOSUB 2150
1770 LOCATE CX,CY,0
1780 '--- DEL
1790 SF=0:BY=LY+CY:TY=CY:LR=LEN(L$(BY))-CX
1800 IF LR=0 THEN 1840
1810 POKE AD,CX+1:IF USR1(L$(BY)) THEN LR=LR-2 ELSE LR=LR-1
1820 C$=RIGHT$(L$(BY),LR):L$(BY)=LEFT$(L$(BY),CX)+C$
1830 PRINT C$;:IF LK(BY)=0 THEN PRINT ES$;"K";:RETURN
1840 IF BY>=EL THEN RETURN
1850 TX=LEN(L$(BY)):IF TX=0 THEN 2020
1860 IF TX<XM OR USR(L$(BY+1))<>1 THEN 1890
```

```
1870  IF TY<YS THEN LOCATE TX,TY:PRINT " ";
1880  LK(BY)=1:RETURN
1890 L$(BY)=L$(BY)+L$(BY+1)
1900 IF TX+LEN(L$(BY+1))>XM THEN 1930
1910  IF TY<YS THEN LOCATE TX,TY:PRINT L$(BY+1);ED$;
1920  LK(BY)=0:BY=BY+1:TY=TY+1:GOTO 2020
1930 POKE AD,XS-TX:IF USR1(L$(BY+1))=1 THEN K=XM ELSE K=XS
1940 IF TY>YM THEN 1960
1950  LOCATE TX,TY:PRINT LEFT$(L$(BY+1),K-TX);SPC(XS-K);
1960 L$(BY+1)=MID$(L$(BY),K+1):L$(BY)=LEFT$(L$(BY),K)
1970 LK(BY)=1:BY=BY+1:TY=TY+1
1980 IF TY<YS THEN LOCATE 0,TY:PRINT L$(BY); ED$;
1990 IF LK(BY) THEN 1840 ELSE RETURN
2000 '--- 行削除
2010 BY=LY+CY:CB$=L$(BY):CK=LK(BY):CX=0:TY=CY
2020 L$(BY)="":LK(BY)=0:IF BY=EL THEN 2050
2030 FOR I=BY TO EL-1:SWAP L$(I),L$(I+1)
2040 SWAP LK(I),LK(I+1):NEXT I
2050 IF TY<YM THEN LOCATE 0,TY:PRINT ES$"M";:GOSUB 3250
2060 EL=EL-1:IF TY>YM THEN RETURN ELSE I=YM:GOTO 3190
2070 '--- 右
2080 S=USR(MID$(L$(LY+CY),CX+1,1))
2090 IF S=1 THEN CX=CX+2 ELSE IF S=0 THEN CX=CX+1
2100 IF S=3 AND CY+LY<EL THEN CX=0:GOTO 2270:RETURN
2110 IF CX<XM-1 THEN RETURN ELSE IF CX>XM THEN 2130
2120 IF LK(LY+CY)=0 OR LEN(L$(LY+CY))>XM THEN RETURN
2130 CX=0:GOTO 2270
2140 '--- 左
2150 IF CX=0 THEN IF LY+CY=0 THEN RETURN ELSE CX=XM:GOTO 2190
2160 POKE AD,CX:IF USR1(L$(LY+CY))=2 THEN CX=CX-2 ELSE CX=CX-1
2170 RETURN
2180 '--- 上
2190 IF CY>0 THEN CY=CY-1:GOTO 2390
2200 IF LY=0 THEN RETURN ELSE LY=LY-1
2210 LOCATE 0,YM:PRINT ES$"M"ES$"H"ES$"L";
2220 I=0:GOSUB 3190:GOTO 2390
2230 '--- 頁上
2240 IF LY=0 THEN RETURN ELSE LY=LY-YM+1:IF LY<0 THEN LY=0
2250 GOSUB 3170:GOTO 2390
2260 '--- 下
2270 IF CY+LY>=EL THEN RETURN
2280 IF CY<YM THEN CY=CY+1:GOTO 2390
2290 LY=LY+1:PRINT ES$"H"ES$"M";:LOCATE 0,YM:PRINT ES$"L";
```

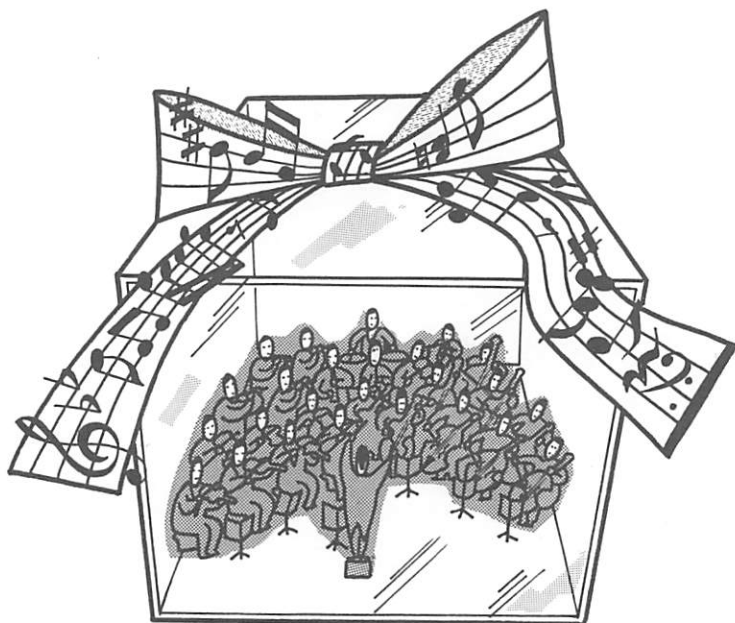
```
2300 I=YM:GOSUB 3190:GOTO 2390
2310 GOSUB 2320:I=YM:GOTO 3190
2320 IF CY<YM THEN CY=CY+1:RETURN
2330 LY=LY+1:PRINT ES$"H"ES$"M";:GOTO 3260
2340 '頁下
2350 IF LY+YM>=EL THEN RETURN ELSE LY=LY+YM-1
2360 IF LY+YM>=EL THEN LY=EL-YM
2370 GOTO 2250
2380 '--- CXの調整
2390 IF CX=0 THEN RETURN
2400 IF LK(LY+CY)AND CX=XM AND LEN(L$(LY+CY))=XM THEN CX=CX-1
2410 POKE AD,CX+1:S=USR1(L$(LY+CY))
2420 IF S=3 THEN CX=LEN(L$(LY+CY)) ELSE IF S=2 THEN CX=CX-1
2430 RETURN
2440 IF CX<LEN(L$(LY+CY)) OR LK(LY+BY)=0 THEN RETURN
2450 CX=0:GOTO 2270
2460 '--- TAB
2470 D=8-(CX MOD 8):IF D>XS THEN D=XS-CX
2480 IF MF THEN C$=SPACE$(D):PRINT C$;:GOTO 2550
2490 CX=CX+D:BY=LY+CY:LL=LEN(L$(BY))
2500 IF CX>=LL AND LK(BY)=0 THEN CX=LL:RETURN
2510 IF CX>XM THEN CX=0:GOTO 2270
2520 POKE AD,CX+1:IF USR1(L$(BY))=2 THEN CX=CX+1
2530 RETURN
2540 '--- 文字列入力
2550 SF=0:BY=LY+CY:TY=CY:LC=LEN(C$):IF MF THEN 2740
2560 '--- 上書
2570 LR=LEN(L$(BY))-CX:IF LC>LR THEN 2620
2580  LR=LR-LC:POKE AD,CX+LC+1:S=USR1(L$(BY))
2590  IF S=2 THEN PRINT " ";:C$=C$+" ":LR=LR-1
2600  L$(BY)=LEFT$(L$(BY),CX)+C$+RIGHT$(L$(BY),LR)
2610  CX=CX+LC:GOTO 2440
2620 L$(BY)=LEFT$(L$(BY),CX)+C$
2630 IF LEN(L$(BY))<XS THEN CX=CX+LC:RETURN
2640 GOSUB 2700:IF LK(BY)=0 THEN 2670
2650 CX=0:IF CY<YM THEN CY=CY+1:GOTO 2550
2660  GOSUB 2310:LOCATE CX,CY:PRINT C$;:GOTO 2550
2670 LK(BY)=1:BY=BY+1:IF CY<YM  THEN CY=CY+1:GOTO 2690
2680  GOSUB 2320:LOCATE 0,CY:PRINT C$;
2690 PRINT ES$;"K";:TY=CY+1:GOSUB 2720:CX=LEN(C$):RETURN
2700 POKE AD,XS-CX:IF USR1(C$)=1 THEN K=XM ELSE K=XS
2710 C$=MID$(L$(BY),K+1):L$(BY)=LEFT$(L$(BY),K):RETURN
2720 GOSUB 3050:I=TY:GOSUB 3190:L$(BY)=C$:LK(BY)=0:RETURN
```

```
2730 '--- 挿入
2740 R$=MID$(L$(BY),CX+1):C$=C$+R$
2750 L$(BY)=LEFT$(L$(BY),CX)+C$:PRINT R$;
2760 IF LEN(L$(BY))<XS THEN CX=CX+LC:RETURN
2770 GOSUB 2700:CX=CX+LC:PRINT ES$;"K";
2780 IF CX>=K THEN CX=CX-K:GOSUB 2320:TY=BY-LY:GOTO 2800
2790  IF CY=YM THEN GOSUB 3260
2800 IF LK(BY)<>0 THEN 2930
2810 LK(BY)=1:BY=BY+1:TY=TY+2:GOTO 2720
2820 '--- CR
2830 IF MF=0 THEN IF BY<EL THEN CX=0:GOTO 2270 ELSE RETURN
2850 '--- 行分割
2860 BY=LY+CY:C$=MID$(L$(BY),CX+1):L$(BY)=LEFT$(L$(BY),CX)
2870 CX=0:PRINT ES$;"K";:IF LK(BY) THEN 2910
2890  BY=BY+1:GOSUB 2310:TY=CY:GOSUB 3050:GOTO 3020
2910 LK(BY)=0:GOSUB 2310:TY=CY-1
2920 '--- 挿入の後始末
2930 TY=TY+1:BY=BY+1:L$(BY)=C$+L$(BY)
2940 IF LEN(L$(BY))>XM THEN 2970
2950  IF TY<YS THEN LOCATE 0,TY:PRINT L$(BY);
2960  LK(BY)=0:RETURN
2970 POKE AD,XS:IF USR1(L$(BY))=1 THEN K=XM ELSE K=XS
2980 C$=MID$(L$(BY),K+1):L$(BY)=LEFT$(L$(BY),K)
2990 IF TY<YS THEN LOCATE 0,TY:PRINT L$(BY);SPC(XS-K);
3000 IF LK(BY) THEN 2930
3010 LK(BY)=1:BY=BY+1:TY=TY+1:GOSUB 3050
3020 L$(BY)=C$:LK(BY)=0:IF TY<YS THEN LOCATE 0,TY:PRINT C$;
3030 RETURN
3040 '--- 画面の下をｽｸﾛｰﾙ
3050 EL=EL+1:L$(EL)="":LK(EL)=0
3060 IF BY<EL THEN 3090
3070  IF TY<YS THEN LOCATE 0,TY:PRINT ES$;"K";
3080  RETURN
3090 FOR I=EL TO BY+1 STEP -1:SWAP L$(I),L$(I-1)
3100 SWAP LK(I),LK(I-1):NEXT I
3110 IF TY<YS THEN LOCATE 0,TY:PRINT ES$"L";:GOSUB 3260
3120 RETURN
3130 '--- 行貼付
3140 TY=CY:BY=LY+CY:CX=0:GOSUB 3050:L$(BY)=CB$:LK(BY)=CK
3150 LOCATE 0,CY:PRINT L$(BY);: RETURN
3160 '--- 全画面表示
3170 FOR I=0 TO YM:GOSUB 3190:NEXT I:GOTO 3250
3180 '--- I行目を表示
```

```
3190 IF I>YM THEN RETURN ELSE LOCATE 0,I,0
3200 IF LY+I>EL THEN 3230 ELSE PRINT L$(LY+I);
3210  IF LEN(L$(LY+I))<XS THEN PRINT ES$;"K";
3220 RETURN
3230 COLOR 11:POKE BC,4:PRINT "~";:COLOR 15,12:PRINT ED$;:RETURN
3240 '--- 状態表示
3250 LOCATE ,,0:IF MF THEN PRINT ES$"y4"; ELSE PRINT ES$"x4";
3260 LOCATE 0,YS,0:COLOR 1:POKE BC,7
3270 IF MF THEN PRINT "<挿入>"; ELSE PRINT "<上書>";
3280 PRINT " [ESC]=ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ";
3290 PRINT ED$;:COLOR 15,12:RETURN
3300 '--- ﾃｷｽﾄを印刷
3310 LOCATE 8,YS:PRINT "[ESC]=ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ";
3320 LOCATE 8,YM:PRINT "何かｷｰを押してください";
3330 A$=INPUT$(1):IF A$=ES$ THEN RETURN
3340 LOCATE 8,YS:PRINT "印刷中です..";ED$;
3350 FOR I=0 TO EL
3360  LPRINT L$(I);
3370  IF LK(I)=0 AND I<EL THEN LPRINT CHR$(&HD);CHR$(&HA);
3380 NEXT I
3390 PRINT "終了.";
3400 RETURN
3410 '--- ﾃｷｽﾄの保存
3420 GOSUB 3780:ON ERROR GOTO 3490
3430 LOCATE 8,YS:PRINT "書込み中です.."ED$;
3440 OPEN FL$ FOR OUTPUT AS #1
3450 FOR I=0 TO EL:PRINT #1,L$(I);
3460  IF LK(I)=0 AND I<EL THEN PRINT #1,CHR$(13);CHR$(10);
3470 NEXT I
3480 CLOSE #1:SF=-1:PRINT "終了.";:RETURN
3490 PRINT "失敗";:LOCATE 8,YM
3500 PRINT "何かｷｰを押してください"ED$;
3510 A$=INPUT$(1):RESUME 3770
3520 '--- ﾃｷｽﾄの読込み
3530 GOSUB 3780
3540 LOCATE 4,YS:PRINT "読込み中.."ED$;
3550 GOSUB 1590:ON ERROR GOTO 3750
3560 OPEN FL$ FOR INPUT AS #1:CLOSE #1
3570 OPEN FL$ AS #1 LEN=128:FIELD #1,128 AS ZZ$
3580 X=0:P$="":AL=LOF(1)
3590 IF AL<=0 THEN 3720 ELSE IF AL>127 THEN I=128 ELSE I=AL
3600 GET #1:AL=AL-I:Z$=P$+LEFT$(ZZ$,I):P$=""
```

```
3610 IL=USR2(Z$):T=PEEK(AD+1):IF IL=0 THEN 3660 ELSE L$(EL)=L$(EL)+LEF
T$(Z$,IL)
3620 IF LEN(L$(EL))<XS THEN 3660
3630  POKE AD,XM:IF USR1(L$(EL))=2 THEN K=XM ELSE K=XS
3640  L$(EL+1)=MID$(L$(EL),K+1):L$(EL)=LEFT$(L$(EL),K)
3650  LK(EL)=1:GOSUB 3730:IF EL=DM THEN 3720 ELSE 3620
3660 IF T=0 THEN 3590 ELSE IF T=1 THEN P$=RIGHT$(Z$,1):GOTO 3590
3670 IF T<>3 THEN 3700
3680  X=LEN(L$(EL)):D=8-(X MOD8):IF X+D>XS THEN D=XS-X
3690  L$(EL)=L$(EL)+SPACE$(D):GOTO 3710
3700 IF T=4 THEN LK(EL)=0:GOSUB 3730:IF EL=DM THEN 3720
3710 Z$=MID$(Z$,IL+2): IF Z$="" THEN 3590 ELSE 3610
3720 CLOSE #1:PRINT "..終了.";:RETURN
3730 EL=EL+1:LOCATE 14,YS:PRINT USING"<###>";EL;:RETURN
3740 'disk ERROR
3750 IF ERL=3560 THEN LOCATE 4,YS:PRINT "新しいﾌｧｲﾙです. ";
3760 RESUME 3770
3770 ON ERROR GOTO 0:RETURN
3780 '--- ﾌｧｲﾙ名入力
3790 LOCATE 8,YS:PRINT "(省略すると '"FL$"' )"ED$;
3800 LOCATE 8,YM:PRINT "ﾌｧｲﾙ名:"ED$;
3810 LINE INPUT D$: IF D$<>"" THEN FL$=D$
3820 LOCATE 15,YM:PRINT FL$;:RETURN
```

MSX2+

ミュージック機能

MSX-MUSICとFM音源

ここでは新しく設定された規格, MSX-MUSICについて, そのベースとなるFM音源のしくみを中心に説明していきます.

MSX-MUSIC

MSXは標準でPSG音源というサウンド機能を装備しています. しかし, このPSG音源では, 好みの音色を作り出すことが難しく, とくにリアルな楽器音などは望めませんでした. このため, ゲームのBGMなどでは, ソフトハウスが独自に音源ICをカートリッジに内蔵したり, 高等テクニックを駆使して対応したりしてきました. しかし, これらの方法は一般ユーザーが自作のプログラムに利用することはできませんでした.

しかし, 以前, MSXにもMSX-AUDIOというFM音源を使ったサウンド機能が, オプション装備としてではありますが用意されました. MSX-AUDIOはFM音源による高品位な音質はもちろんのこと, よりリアルなサウンドが楽しめるサンプリングができたり, 鍵盤を付ければ本格的なシンセサイザーにもなるという高機能なものでした. ところが, 単純にゲーム等でFM音源によるBGMを付けたりするような用途にはちょっと高価なものだったため, あまり一般のユーザーには受け入れられなかったようです.

そこで, もっと機能を絞り込んで, 一般のユーザーが手軽にFM音源を使用できるように, 新たに用意されたのが, MSX-MUSICです. これは, 松下電器から発売されているパナ・アミューズメント・カートリッジ(FM-PAC)のFM音源機能と基本的に同じものです.

■ MSX-MUSICの特徴

MSX-MUSICでは，音源ICとして，ヤマハのOPLL(YM2413)というFM音源ICが使われています．このICは，MSX-AUDIOで使われていた，OPL(Y8950)から，ADPCMサンプリングなどの機能を取り除き，機能をFM音源に絞り込んだICです．

このICの採用により，MSX-MUSICでは，メロディ音を9音または，メロディ音6音＋リズム音5音の2つの発音モードを持っていて，この2つのモードをソフトウェアで切り替えることができます．そして，両モードとも，同時に違う音を発音することが出来ます．

このOPLLは，内部に音色ROMを搭載しており，そのなかにあらかじめ15音色分のデータが登録されているため，従来のFM音源ICでは非常に面倒だった音色の設定も，音色番号を指定するだけで設定できるようになっています．

もちろん，効果音や独自の音を作って使うことができるように，1音色分のオリジナル音色レジスタも用意されています．

MSX-MUSICのFM音源も従来のPSG音源と同じように，BASICからPLAY文で制御することができます．

FM音源の発音の仕組み

FM音源は，リアルな楽器音や独自に創り出した複雑な音を，比較的簡単に鳴らすことができます．PSGでは難しかったリアルな音作りを可能にしているFM音源とは，いったいどのようなものでしょうか．

■ 音の仕組み

FM音源音を知るためには，まず音そのものの仕組みを知っておく必要があります．ここでは簡単に，音の仕組みについて触れておきましょう．

音とはいったいどういうものかを表すには，一般的に，音の大きさ，音の高さ，音色，という3つの要素を使います．音は，これらの3要素が複雑に絡み合ってできているということができます．これらのうち，音の大きさや，音の高さに関しては誰でも感覚的に理解できるものですが，音色については，

ちょっと感覚的に理解することは難しいでしょう．

そこで，音を視覚的に捉えてみることにしましょう．音を電気信号に直すと，波形で表わすことができます．波形として見た場合，音の高さは波の細かさ(周波数の高さ)にあたり，音の大きさは波の振幅(ゆれ幅)の大きさに当ります(図 4.1)．

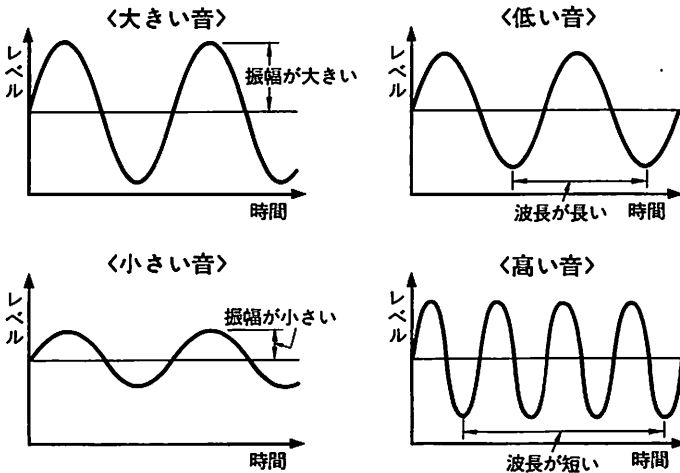



図 4.1　音の性質と波形

普通，人間の耳に聞こえる音は，その音(基音)とその音の何倍かの周波数(倍音)をいくつも含んでいます．基音の何倍かの周波数ということは，音程の高い音ということになり，それならば和音として聞こえそうなものですが，整数倍の倍音は基音とよく混じるため，人間の耳には単音に感じられます．

この周波数成分(倍音)の含まれ方によって，音色が決定されているのです．ここでは簡単に，音色は，「その音の周波数成分の含まれ方」ということにします．

さて，さきほど音は，音の高さ，大きさ，音色で構成されていると説明しましたが，もう1つ，音は時間がたつにつれて変化していきます．したがっ

て，さきほどの3要素に加え，その音の時間的な変化も考えなければなりません．そこで，音を決定するための要素は，

音の高さの時間的変化
音の大きさの時間的変化
周波数成分の時間的変化

という3つであるということができます．

つまり，ピアノの音を構成するこれら3要素を，なんらかの方法でそっくりに合成することができれば，ピアノの音を創り出せるということになります．

■ オペレータの機能

さて，音の基本を理解したところで，FM音源の発音の仕組みを詳しく見ていきましょう．

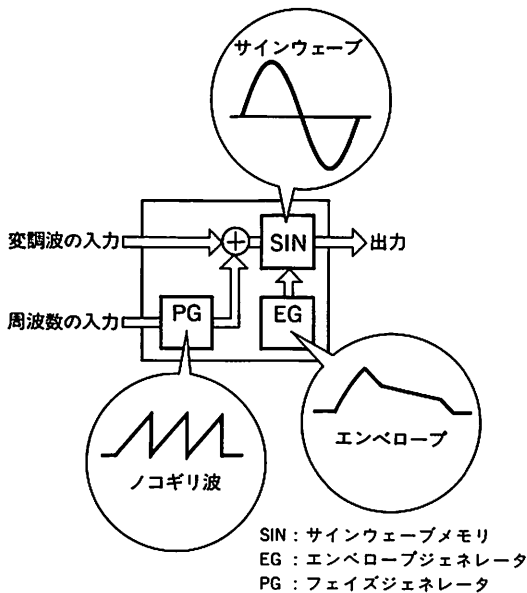



図4.2　オペレータの基本構成

FM音源の基本発音要素はオペレータといわれる発振器です．オペレータはサインウェーブメモリ(SIN)，フェイズジネレータ(PG)，エンベロープジェネレータ(EG)の3つの装置から構成されています(図4.2)．

サインウェーブメモリのなかには，その名のとおり1サイクル分の正弦波をデジタルに変換したデータが納められています．

フェイズジェネレータはそのオペレータの基本周波数を決めるためのもので，つねに与えられた周波数に対応するノコギリ波を出力しています．

サインウエーブメモリのなかに納められたデータを，フェイズジェネレーターからのノコギリ波に合わせて読み出すと，図4.3のように正弦波を出力することができます．つまり，のこぎり波1サイクルでサインウェーブ1サイクルが読み出されるようになっています．

たとえば，フェイズジェネレータに与える周波数を高くして，サインウェーブメモリを読み出す速度を速くすれば，出力される波形の周波数も高くなり，反対に，フェイズジェネレータに与える周波数を低くすれば，出力される波形の周波数も低くなります．

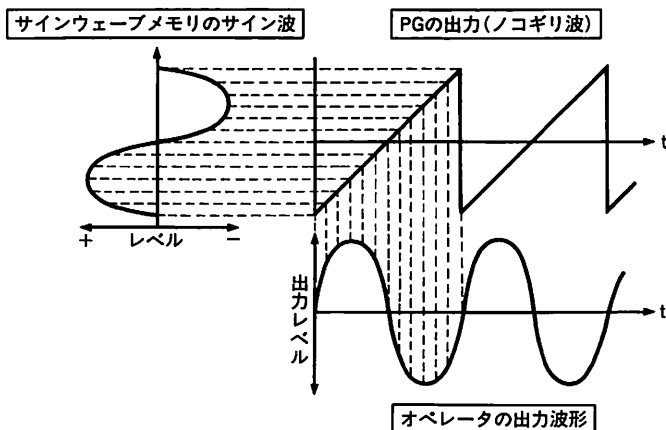



図4.3 サインウェーブメモリの読み出し

エンベロープジェネレータは，上記の方法によって作られた音に，時間的な音量の変化を付けるためのものです．たとえばピアノの，鍵盤を押したときに一番音が大きく，押し続けているとだんだん音が小さくなる，というような音量の変化を電気的に再現するものです．

楽器の音にはいろいろな音量変化のパターンがあるのですが，代表的なものとして，減衰していく音と，ある一定のレベルで持続する音に分けられます．OPLL ではこの 2 つのパターンをシュミレートすることができます．そしてこの 2 つのパターンの音の時間的な音量変化を 4 つのパラメータで表現しています．パラメータはそれぞれ，

・持続音の場合

①アタック・レイト	音の鳴り初めから音量が最大になるまでの時間
②ディケイ・レイト	最大からサスティン・レベルになるまでの音量の減衰率
③サスティン・レベル	音量が落ち付くレベル
④リリース・レイト	キーを離してから音量が 0 になるまでの時間

・減衰音の場合

①アタック・レイト	音の鳴り初めから音量が最大になるまでの時間
②ディケイ・レイト	最大からサスティン・レベルになるまでの音量の減衰率
③サスティン・レベル	ディケイ・レイトからリリース・レイトへ変化する音量
④リリース・レイト	サスティン・レベルからの音量の減衰率

を表しています．また，図 4.4 の Key　ON／OFF はそれぞれ，鍵盤を押した場合と離した場合に相当します．

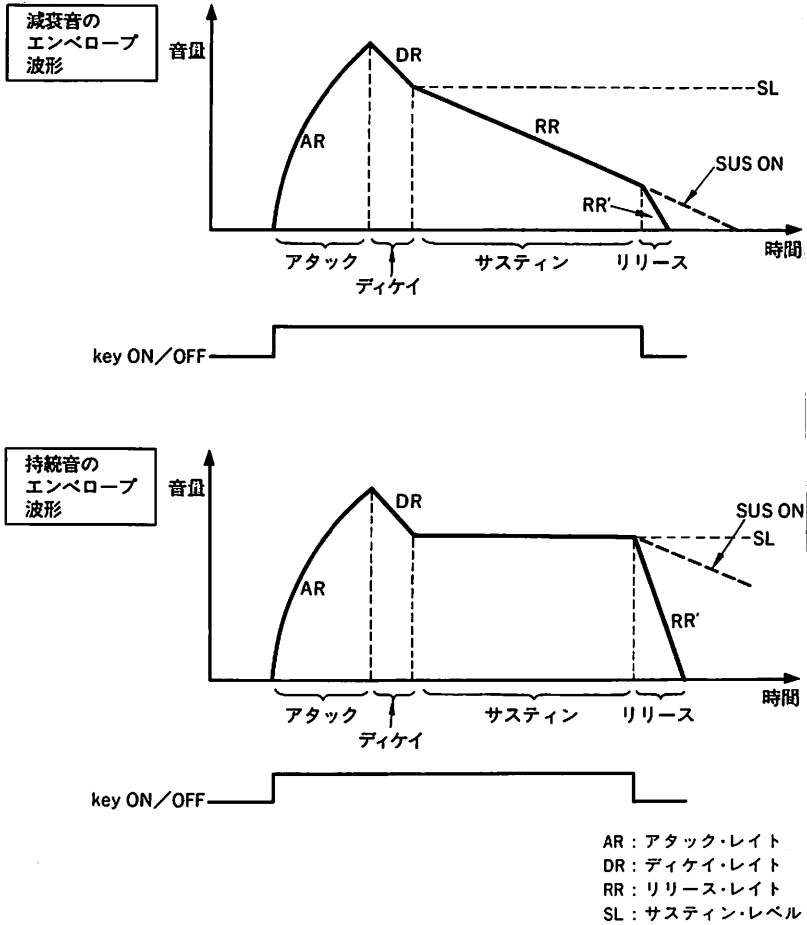



図 4.4　エンベロープパターン

■ 周波数合成

さて，オペレータ1つでは，ただの正弦波発生装置でしかありません．音色を作るためには，いろいろな周波数成分が合成された波形を作らなければなりません．そこで，オペレータを2つつなげて使います(図 4.5)．

この2つのオペレータを区別するために，オペレータ1のことをモジュレータ，オペレータ2のことをキャリアと呼びます．

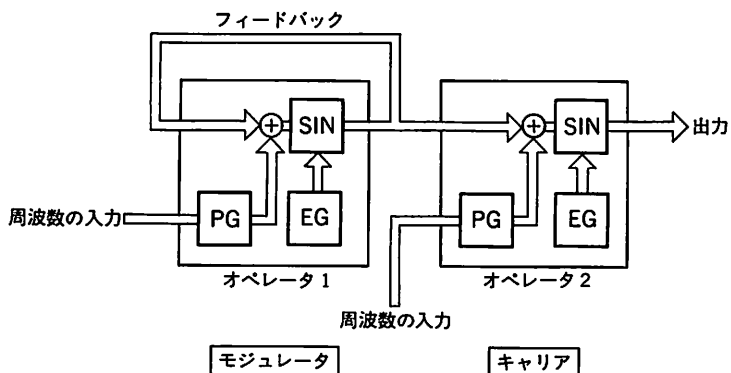



図 4.5　2 つのオペレータをつなげる

まず，モジュレータで作られた出力と，キャリアのフェイズジェネレータの出力とをたし合わせます(図 4.6).

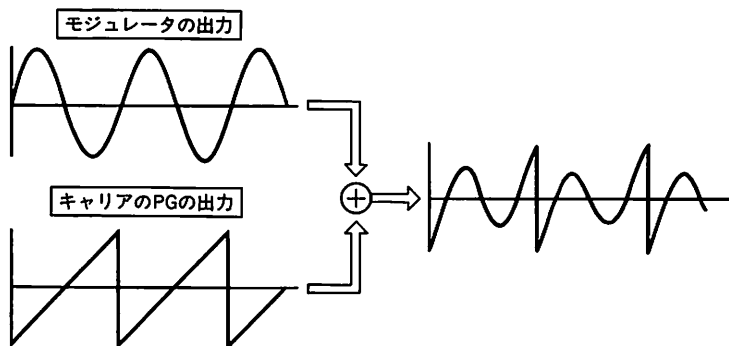



図 4.6　モジュレータの出力とキャリアの PG の出力を合成する

合成された信号でキャリアのサインウェーブメモリを読み出します．そうすると，サインウェーブメモリを読み出すタイミングが変化して，いろいろな倍音成分を含んだ波形ができあがります(図 4.7)．つまり，モジュレータの出力で，キャリアの周波数に変化を加えてキャリアを変調するわけです．

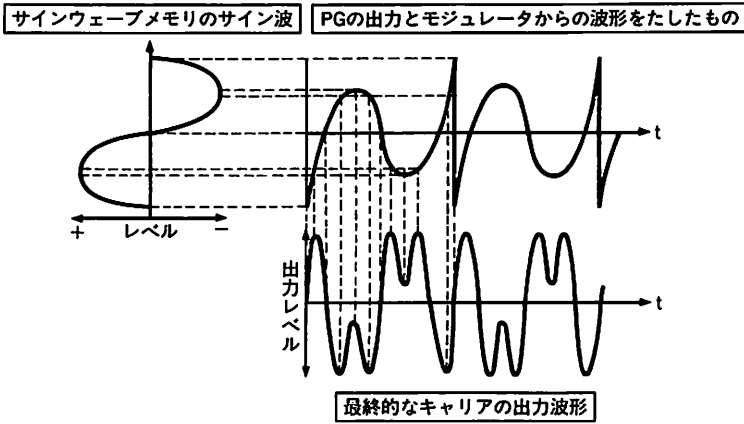



図 4.7 キャリアでのサインウェーブメモリの読み出し

モジュレータ側の出力の音量変化(エンベロープパターンによる音量変化)は，キャリアの出力する音の，周波数成分の音量の時間的変化となって現れます．

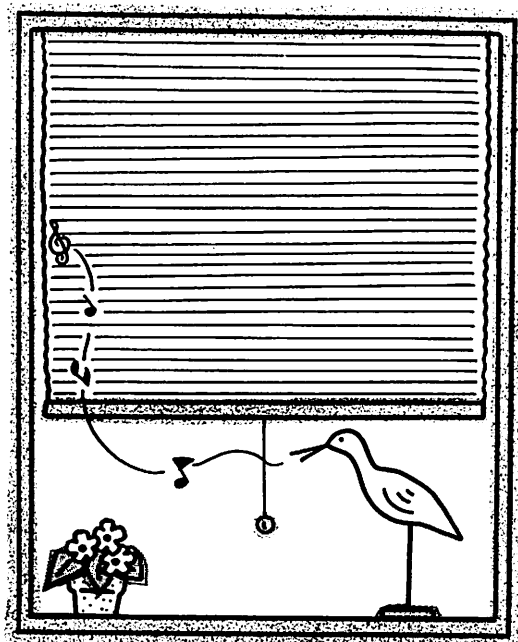
モジュレータ側のフェイズジェネレータの周波数は，キャリアの出力する音の周波数成分(倍音)の含まれ方に影響します．キャリア側のフェイズジェネレータの出力は，キャリアから出力する音の高さに影響します．

キャリア側のエンベロープジェネレータは，最終的にできあがった音の大きさを制御します．

この調子で，オペレータをたくさんつないでいけば，もっと多くの周波数成分を含んだ信号ができあがりますが，自分自身の出力を自分に加えることによっても，同じような効果を出すことができます．これをフィードバックと呼びます．フィードバックの量が増えれば，含まれる周波数成分が多くなって，鋭い音になります．

このように，モジュレータ，キャリアのフェイズジェネレータに設定する周波数や，エンベロープパターンの設定，モジュレータのフィードバックの量を変化させることによって，「音の高さの時間的変化」，「音の大きさの時間的変化」，「周波数成分の時間的変化」を自在に作り出せ，複雑な音を再現できるわけです．

オペレータの数を増やしていけば，もっとたくさんの変化を付けることができるのですが，残念ながらOPLLではオペレータの数が1音につき2オペレータになっていて，その2つのオペレータのつなぎ方も直列に固定されているため，創り出せる音色もおのずと限られてきます．

しかし，2オペレータでも，かなり複雑な波形を出力できますので，ゲームや自作のプログラムのBGMなどには十分なものとなっています．

2 MSX-MUSIC拡張BASICコマンドリファレンス

MSX-MUSICのFM音源をBASICから制御するための拡張BASICのコマンドを解説していきます．これは，松下から発売されているパナアミューズメントカートリッジ(FM-PAC)に内蔵されている拡張BASICと同じものです．

この拡張BASICは，今までのPSG音源とOPLLのFM音源を同時に制御することが可能になっています．ですから，FM音源9音＋PSG音源3音の12音を同時に発声することができます．これまで使われてきたPSG用のMML(ミュージックマクロランゲージ)と文法は基本的に同じですから，これまでに作られてきた，PSG音源用のプログラムを多少の変更でFM音源用のMMLとして使うことができます．

また，MSX-AUDIOの拡張BASICとの互換性がありますので，MSX-AUDIO用に作られたプログラムも多少の変更でMSX-MUSICでも使うことができるようになります．MSX-AUDIOのコマンドで，MSX-MUSICで使えないコマンドについては，本章末で説明します．

MSX-MUSIC拡張BASICのコマンドには以下のものがあります．

CALL　MUSIC
AUDREG
BGM
PITCH
PLAY
STOPM
TEMPER
TRANSPOSE
VOICE
VOICE　COPY

MUSIC

●書式● CALL MUSIC [([<モード>] [, [0]
[, <PLAY 文第 1 文字列へのチャンネル数>
[, <PLAY 文第 2 文字列へのチャンネル数>
[, …… [, <PLAY 文第 9 文字列へのチャンネル数>]
……]]]])]

MSX-MUSIC のシステムを初期化します．パラメータとして，9 個の FM 音源のチャンネルをどのように使用するかを指定します．

MUSIC 文により，初期化を行うまでは，すべての拡張 BASIC ステートメントは使用できません．

<モード>は 0 か 1 で，0 はリズム音を使用しないモードで，1 はリズム音を使用するモードです．リズム音を使用する場合にはチャンネル 7，8，9 をリズム音用に使用しますので，メロディ音に使えるのは残りの 6 チャンネルになります．したがって，PLAY 文で使用するチャンネル数はリズム音使用時は 6 以下，リズム音を使用しないときは，9 以下である必要があります．

チャンネルの使用割当は，PLAY 文では，チャンネル番号の小さい方(1，2，3……)から割り当てます．

パラメータを 1 つ以上指定した場合，他のパラメータの省略時の値は 0 となります．

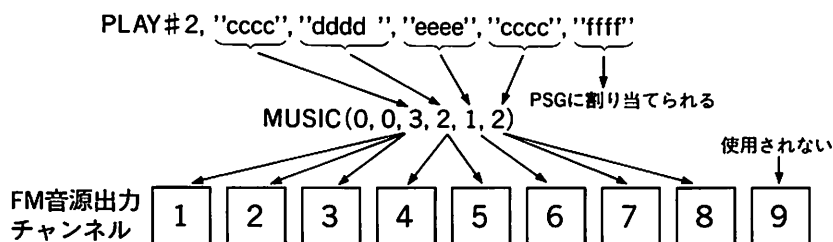



図 4.8 MUSIC 文のチャンネル割当

PLAY 文の文字列へのチャンネル数に，0 を指定したり省略したりすることはできません．以下の例を参照してください．

・0 を設定した場合
CALL MUSIC(0,0,0,5,0)
この場合〝Illegal function call〟になる

・省略した場合
CALL MUSIC(0,0,1,,2)
この場合〝Syntax error〟になる

パラメータをすべて省略したときは，

CALL MUSIC(1,0,1,1,1)

という設定をしたのと同じになります．すなわち，

・FM 音源のチャンネル 1 を PLAY 文の最初の文字列に割り当てる
・FM 音源のチャンネル 2 を PLAY 文の 2 番目の文字列に割り当てる
・FM 音源のチャンネル 3 を PLAY 文の 1 番目の文字列に割り当てる
・リズム音を PLAY 文の 4 番目の文字列に割り当て使用する
・PLAY 文の 5 番目以降の 7 番目までの文字列は PSG 音源の制御に割り当てる

という意味になります．

MUSIC 文を実行すると，システムの割り込みのフックが MSX-MUSIC のシステムソフトウェアにリンクされるので，割り込み処理ルーチンのオーバーヘッドが増え，システムのスループットが低下します．また，MUSIC 文はワークエリア保護のために内部で CLEAR 文に相当することを行っているので，HIMEM(CLEAR 文の第 2 パラメータに相当します)が 807 バイト小さくなり，変数はすべてクリアされます．

AUDREG

●書式●　CALL AUDREG(〈レジスタ番号〉, 〈値〉 [, 〈チャンネル番号〉])

音源チップのレジスタに直接値を書き込みます．レジスタ番号，レジスタの機能の内容については，次章のレジスタ表を参考にしてください．

どのレジスタにも値を書き込むことができますが，システムソフトウエアが割り込みなどで頻繁に書き込んでいるレジスタには，効果がない場合や，システムの再立ち上げが必要な場合があります．

〈チャンネル番号〉は，省略するか，0を指定しなければなりません．この〈チャンネル番号〉は，MSX-AUDIO 拡張 BASIC との互換性をとるために用意されているもので，MSX-MUSIC では，意味を持ちません．

リスト4.1　音色の設定をレジスタ直接書き込みで行う

```
100 CALL MUSIC
110 CALL VOICE COPY(@1.@63)
120 CALL VOICE(@63)
130 PLAY #2,"O4V15CDEFGABO5C"
140 CALL PLAY(0,A):IF A=-1 THEN 140
150 CALL AUDREG(0,&B11001111)
160 PLAY #2,"O4V15CDEFGABO5C"
```

BGM

●書式●　CALL BGM(〈変数〉)

PLAY 文の実行をバックグラウンド処理として実行するかどうかを指定します．

音楽の処理では，ゲーム等のプログラムで，BASIC の処理をしながら音楽を鳴らすことができるようになっています．このことをバックグラウンド処理といいます．

PLAY 文では, MSX の 60 分の 1 秒ごとにかかる割り込み信号を使って, BASIC プログラムの実行と PLAY 文の実行を切り替えながら処理しています. バックグラウンド処理を禁止すると, PLAY 文の実行が終わるまで BASIC プログラムの実行がされなくなります.

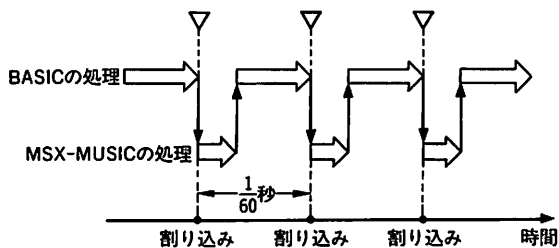



図 4.9 バックグラウンド処理の概念図

〈変数〉は 0 または 1 の値をとります.

CALL BGM(0)

と指定すると, PLAY 文のバックグラウンド処理を行いません.

CALL BGM(1)

と指定すると, PLAY 文のバックグラウンド処理を行います.

PLAY 文による初期化では, バックグラウンド処理が指定されていますが, 〈変数〉に 0 を与えることで, フォアグラウンド処理にすることができます.

リスト 4.2 バックグラウンド処理を行う

```
100 CALL KANJI1:CALL MUSIC:SCREEN 8
130 CALL BGM(1)
140 COLOR 255:PRINT "バックグラウンド処理をしています。"
150 GOSUB 200
160 CALL BGM(0)
170 COLOR 255:PRINT "バックグラウンド処理をしていません。"
```

```
18Ø GOSUB 2ØØ
19Ø END
2ØØ PLAY #2,"@16@V1ØØO5","@13@VØ95O4","@Ø8@V1ØØO6"
22Ø PLAY #2,"G8F+8G4F+8D8F+4L8GCEGF+DF+4","G2D2C2D2"
23Ø GOSUB 3ØØ
24Ø PLAY #2,"G8F+8G4F+8D8F+4L8GCEGBAG4","G2D2C2<G2>"
25Ø GOSUB 3ØØ
26Ø PLAY #2,"G8F+8G4F+8D8F+4L8GCEGF+DF+4","G2D2C2D2","L8GDBDF+DAF+CEGE
<A>DF+D"
27Ø GOSUB 3ØØ
28Ø PLAY #2,"G8F+8G4F+8D8F+4L8GCEGBAG4","G2D2C2<G2>","L8GDBDF+DAF+CEGE
<B>DG4"
285 GOSUB 3ØØ:RETURN
3ØØ FOR I=1 TO 13
3Ø5 X=INT(RND(1)*255):Y=INT(RND(1)*255):C=INT(RND(1)*255)
31Ø LINE (X,Y)-(Y,X),C,BF
32Ø NEXT I:RETURN
```

PITCH

●書式●　CALL PITCH(〈ピッチ1〉 [,〈ピッチ2〉])

FM音源で発生する楽音の音高(ピッチ)を指定します.〈ピッチ〉の範囲は410～459で単位は [Hz] です.中央Cのすぐ上のA音の周波数で音高を表します.トランスポーズとは独立に設定でき,デフォルトの値は440です.

ピッチ(または,トランスポーズ値)を変えるとリズム音や音程を持たない音を除くFM音の音の高さが変化します.PSG音源には作用しないので注意してください.

ピッチ2は指定しても無視されます.これはMSX-AUDIO拡張BASICとの互換性をとるために用意されているものでMSX-MUSICでは意味を持ちません.

トランスポーズについては,TRANSPOSE文の項を参照してください.

リスト 4.3 順次ピッチを変えて演奏する

```
100 CALL MUSIC
105 FOR I=0 TO 49
110 CALL PITCH(410+I)
120 PLAY #2,"L15V15A"
130 NEXT I
```

PLAY

●書式● PLAY [# 〈モード〉,] 〈文字列 1〉 [, 〈文字列 2〉 [, 〈文字列 3〉 … [, 〈文字列 13〉]]]

PLAY 文は音楽を演奏するもので，FM 音源 9 音と従来の PSG 音源 3 音の最大 12 声まで同時発声が可能です．〈文字列 n〉に書かれたミュージックマクロランゲージ(MML)が演奏されます．他の拡張命令と異なり CALL 文は必要ありません．

〈モード〉は 0 から 3 までの値をとり，PLAY 文の音源や動作モードを次のように設定します．

0 か省略されたときは PSG のみが音源となり，文字列は最大 3 つまでとなります．従来の PLAY 文と互換性があります．

1 のときは，“Illegal function call” となります．これは MSX-AUDIO 拡張 BASIC との互換性をとるために用意されているもので，MSX-MUSIC では意味を持ちません．

2 または 3 のときは，FM 音源，リズム音，PSG 音源を使用できます(2 のときと 3 のときで動作には違いはありません)．

〈文字列〉と音源の関係は，CALL MUSIC 文で設定された MML の個数に対応して，

〈FM音源用文字列1〉，…，〈FM音源用文字列n〉，
〈リズム音用文字列〉，〈PSG音源用文字列1〉，
〈PSG音源用文字列2〉，〈PSG音源用文字列3〉

となります．また，MUSIC文でリズム音を使用しないモードに設定した場合は，リズム音用文字列をカンマとともに省略しなければなりません．

例として，デフォルトのMUSIC文の設定に対する文字列の配列をあげると，次のようになります．

PLAY #2，〈FM音源用文字列1〉，〈FM音源用文字列2〉，
〈FM音源用文字列3〉，〈リズム音用文字列〉，
〈PSG音源用文字列1〉，〈PSG音源用文字列2〉，
〈PSG音源用文字列3〉

MSX-MUSICではFM音源用に従来のMML（ミュージックマクロランゲージ）に若干の拡張をしています．FM音源用に新設されたMMLについて説明します．そのほかのコマンドについては従来と同じです．

Qn （1≦n≦8）

音の長さの設定で決められる長さ（ステップタイム）と，その間に実際に音を出している長さ（ゲートタイム）との比を設定します．nの値は1～8で，たとえば4に設定すると，ゲートタイムの長さは，ステップタイムの長さの4/8になります．デフォルトでは8に設定されています．PSG音源用文字列のなかで使用した場合は無視されます．

{} n （1≦n≦64）

連符を表します．nで指定された音長を{}のなかの音符の個数で等分して，{}のなかの音符の音程で，その長さの音を発声します．{}のなかには音符，休符以外の文字を指定することはできません．このコマンドはFM音源に対してのみ有効で，PSG音源用文字列のなかで使用するとエラーとなります．

@n （0≦n≦63）

文字列と対応しているFM音源のチャンネルの音色を，nで指定する音色に切り換えます．音色番号については，VOICE文の項を参照してください．

音色番号表のなかで OPLL VOICE の欄に指定のない音色については，同時に 1 音色しか設定できません．もし，OPLL VOICEの指定のない音色が複数指定された場合は，いちばん最後に指定した音色が採用されます．PSG 音源用文字列のなかで使用した場合は無視されます．

@Vn　(0≦n≦127)

FM 音源に対する音量を細かく設定します．FM 音源では PSG よりも細かく音量を設定することができます．PSG 音源用文字列のなかで使用した場合は無視されます．

@Wn　(1≦n≦64)

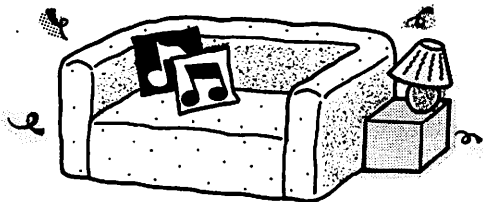
n で指定した長さだけ何もしないで待ちます．このコマンドは次に解説する Yr, d コマンド等で，音を発声させる場合に有効です．Rn コマンドでは，発音をいったん停止してしまいますが，このコマンドでは本当に何もしません．

PSG 音源用文字列のなかや，CDE 等の音符のあとに使った場合は，Rn コマンドと同様です．

Yr，d

OPLL のレジスタに直接値を書き込みます．どのレジスタにも値を書き込むことができますが，システムソフトウェアが割り込みなどで頻繁に書き込んでいるレジスタには，効果がない場合や，システムの再立ち上げが必要な場合があります．

PSG 音源用文字列のなかで使用した場合は無視されます．

その他の MML コマンドについては次の一覧表を参照してください．

文　字	意　　味	値のとる範囲	デフォルト	FM音源	PSG音源
Mn	エンベロープ周期の設定	1≦n≦65535	M255	—	○
Sn	エンベロープ形状の設定	0≦n≦15	S0	—	○
Vn	音量の設定	0≦n≦15	V8	○	○
Ln	長さの割合	1≦n≦64	L4	○	○
Qn	音の長さの割合	1≦n≦8	Q8	○	—
On	オクターブの設定	1≦n≦8	O4	○	○
>	オクターブを1つ上げる			○	○
<	オクターブを1つ下げる			○	○
Tn	テンポの設定	32≦n≦255	T120	○	○
Nn	nで指定された高さの音を発生する	0≦n≦96		○	○
Rn	休符の設定	1≦n≦64	R4	○	○
A～G	音程の発生			○	○
+, #	音を半音上げる			○	○
−	音を半音下げる			○	○
.(ピリオド)	音符や休符の長さを1.5倍する			○	○
=x;	パラメータnを変数xで設定する			○	○
Xx;	文字変数xに入っているMMLを演奏する(注)			○	○
&	タイ，前後の音をつなぐ			○	○
{ }n	連符，n分音符を{ }の中の音程の個数で等分にした音を発生する	1≦n≦64	Lnで設定された値	○	×
@n	n番の音色に切り替える	0≦n≦63		○	—
@Vn	音量を細く設定する	0≦n≦127		○	—
@Wn	nで指定された長さだけ状態を継続する	1≦n≦64	Lnで設定された値	○	Rnと同等
Yr, d	音源チップのレジスタrにdを書き込む			○	—
Zd	MSX-AUDIOではMIDIポートへの出力だが，MSX-MUSICではエラーになる			×	—

注：このマクロを指定した場合，このマクロ以降に何かマクロを書くことはできません，書いた場合はエラーとなります．
○……使用可　×……使用不可　—……無視

表 4.1　MML の仕様と各音源への対応

リズム音源に対する MML は，1 つの MML で同時に複数の音を発声するために，楽音用のものとは違った記述様式をとります．

まず鳴らしたい楽器の文字を並べて，そのあとに長さを指定します．

文　字	意　　味	値のとる範囲
B	バスドラム音を発生	
S	スネアドラム音を発生	
M	タムタム音を発生	
C	シンバル音を発生	
H	ハイハット音を発生	
!	直前の楽器の音量をアクセントボリュームにする	
n	直前までに書かれた楽音を発生し、n 分音符分待つ	1 ≦ n ≦64
Vn	アクセントの付いていない楽音の音量を設定する	0 ≦ n ≦15 デフォルト = 8
@An	アクセントの付いている楽音の音量を設定する	0 ≦ n ≦15

Tn, @Vn, Rn, =x;, Xx;, . はFM音源用と同様です.

表 4.2　リズム用 MML の仕様

リズムの指定は以下のように行います．

"BSH8H8S!H8H8"

この命令の意味は，

- バス，スネア，ハイハットを鳴らし，8 分音符分待つ
- ハイハットを鳴らし，8 分音符分待つ
- スネアをアクセント付きでハイハットと鳴らし，8 分音符分待つ
- ハイハットを鳴らし，8 分音符分待つ

ということになります．

リスト 4.4　PLAY 文による演奏

```
100 CALL MUSIC(1,0,2,2,2)
110 PLAY #2,"@24T150","@12T150","@21T150","T150"
120 PLAY #2,"L804AE<A>EAE<A>EA<EA>AA<EA>AAE<A>EAE<A>EA<EA>AA<EA>A","03
A1.A4G4F1.F4E4","R1R1R1R1","CB4S4B4S4B4S4CB4SCB4CB4S4B4S4B4S4CB4SCB4"
130 PLAY #2,"L804AE<A>EAE<A>EA<EA>AA<EA>AAE<A>EAE<A>EA<EA>AA<EA>A","D1
.D4E4<A1>","R1R1R1R1","CB4S4B4S4B4S4CB4SCB4CB4S4B4SB4B4S4B4SCB16S16C16
"
140 PLAY #2,"L804AE<A>EAE<A>EA<EA>AA<EA>AAE<A>EAE<A>EA<EA>AA<EA>A","03
A1.A4G4F1.F4E4","L8AAAA>CCCC<AAAA>CCEE<AAAA>CCCC<AAAA>CCEE<","CB4S4B4S
4B4S4CB4SCB4CB4S4B4S4B4S4CB4SCB4"
150 PLAY #2,"L804AE<A>EAE<A>EA<EA>AA<EA>AAE<A>EAE<A>EA<EA>AA<EA>A","D1
.D4E4<A1>","L8DFDADFDADFA>DFD<AF<AAAAAAAA>AAAAAAAA","CB4S4B4S4B4S4CB4S
CB4CB4S4B4SB4B4S4B4SCB16S16C16"
160 GOTO 120
```

PLAY(　)

●書式●　CALL PLAY(〈PLAY 文のストリング番号〉, 〈変数名〉)

PLAY 文が音楽を演奏中かどうかを調べます．PLAY 文のミュージックキューの状態を調べ，各チャンネルが音楽を演奏中かどうか判断し，演奏中であれば-1 を，そうでなければ 0 を〈変数〉に代入します．ただし，〈ストリング番号〉として 0 が与えられた場合は，いずれかのストリングが演奏中であれば-1 を，そうでなければ 0 を〈変数〉に代入します．

PLAY 文のストリング番号は，MUSIC 文で指定したストリング数+3 まで使えます．すなわち，MUSIC 文で指定した FM 音源に加え，3 チャンネルの PSG 音源に関しても演奏中かどうかの判定ができます．

STOPM

●書式●　CALL STOPM

バックグラウンドで実行中の PLAY 文の音楽の演奏を強制的に停止させます．

リスト 4.5　演奏を中断させる

```
100 CALL KANJI1
110 CALL MUSIC
120 ON KEY GOSUB 200:KEY(1) ON
130 PRINT "とめるときはＦ１キーを押してください"
140 PLAY #2,"@16@V10006","@04@V09505","@05@V09505"
150 PLAY #2,"L8GDBDF+DAF+CEGE<A>DF+D","G2D2C2D2"
160 PLAY #2,"L8GDBDF+DAF+CEGE<B>DG4","G2D2C2<G2>"
170 PLAY #2,"L8GDBDF+DAF+CEGE<A>DF+D","G2D2C2D2","B2F+2E2F+2"
180 PLAY #2,"L8GDBDF+DAF+CEGE<B>DG4","G2D2C4D8E8G2","B2F+2E4D8C8<B2"
190 GOTO 140
200 CALL STOPM:RETURN 210
210 END
```

TEMPER

●書式●　CALL TEMPER(〈音律番号〉)

音律を与えるステートメントで，FM 音源の楽音の音高に影響を与えます．〈音律番号〉には 0 から 21 の音律番号を指定します．デフォルト値は 9 番の完全平均率です．

音律は 1 オクターブ 12 音をどのような比率で分割するかを決めるもので，古典音楽には古典音律が適しているといわれます．音律を変更すると，それぞれの音の周波数のずれによって，和音の響き方が違ってきます．

番号	音　　　律	
0	ピタゴラス	
1	ミーントーン	
2	ヴェルクマイスター	
3	ヴェルクマイスター(修正)	
4	ヴェルクマイスター(別)	
5	キルンベルガー	
6	キルンベルガー(修正)	
7	ヴァロッティ・ヤング	
8	ラモー	
9	完全平均律(デフォルト)	
10	純正律　c	メジャー(aマイナー)
11	純正律　cis	メジャー(b)
12	純正律　d	メジャー(h)
13	純正律　es	メジャー(c)
14	純正律　e	メジャー(cis)
15	純正律　f	メジャー(d)
16	純正律　fis	メジャー(es)
17	純正律　g	メジャー(e)
18	純正律　gis	メジャー(f)
19	純正律　a	メジャー(fis)
20	純正律　b	メジャー(g)
21	純正律　h	メジャー(gis)

(修正)：平島達司氏による

表 4.3　音律表

リスト 4.6　音律を変えながら演奏

```
100 CALL MUSIC(0,0,3)
110 CALL BGM(0)
120 FOR I=0 TO 21
130 CALL TEMPER(I)
140 PRINT I
150 PLAY #2,"@0L4O4V15"
160 PLAY #2,"CDEFGABO5C"
170 NEXT I
```

TRANSPOSE

●書式● CALL TRANSPOSE(〈トランスポーズ値 1〉
[, 〈トランスポーズ値 2〉])

FM 音源の楽音に対してセント単位で移調を行います．セントとは半音を100 とした移調の単位で，1 オクターブ上げるには，+1200 を与えます．

トランスポーズ値として許される値の範囲は±12799 以内ですが，実際には FM 音源の音色によって，ある高さの範囲以外は制限されます．音高精度は LSI の制限により±2 セント程度です．

トランスポーズはピッチとは独立して設定できます．MUSIC 文による初期化の値は 0 です．ピッチについては，PITCH 文を参照してください．

〈トランスポーズ値 2〉は指定しても無視されます．これは MSX-AUDIO 拡張 BASIC との互換性をとるために用意されているもので，MSX-MUSIC では意味を持ちません．

リスト 4.7　半音ずつ移調しながら演奏

```
100 CALL MUSIC
110 CALL BGM(0)
120 FOR I=0 TO 1200 STEP 100
130 CALL TRANSPOSE(I)
140 PLAY #2,"T120@15@V10505","T120@13@V09504"
150 PLAY #2,"L8GDBDGDBDF+DAF+F+DAF+","L4GGGGDDDD"
160 PLAY #2,"L8GE<B>EGECEF+DAF+F+DAF+","EECCDDDD"
170 NEXT I
```

VOICE

●書式● CALL VOICE([<チャンネル1用のボイス>],
[<チャンネル2用のボイス>], ……,
[<チャンネル9用のボイス>])
(ボイス=@+数式　または　配列変数名)

9チャンネルあるFM音源のそれぞれに音色を設定します．音色の設定方法には2種類の方法があります．1つはシステムに備えられている音色ライブラリを使う方法で，0から63の音色の番号を数式により指定します．この場合には数式の前に@記号を付けて次の配列変数名と区別します．

プログラムにより音色パラメータを与えて設定する場合には，配列変数に音色パラメータを入れてその配列変数名を指定します．音色パラメータのフォーマットの詳細はVOICE COPY文の項を参照してください．パラメータを省略したチャンネルの音色は変更されません．

表4.4の音色ライブラリのうち，OPLL VOICEの欄に指定のあるものはOPLL内蔵の音色に対応しています．それ以外の音色はオリジナル音色に当たります．OPLLでは，オリジナル音色は1音色分しか設定できませんので，これらの音色や，自分で作った音色を配列変数で設定する場合には，同時に1音色しか設定できません．複数設定しようとした場合には，パラメータ列のいちばん左側のパラメータか最後に実行したCALL　VOICE文の設定のみが有効となります．いくつかの例を以下に示します．

```
CALL VOICE(@26, @27)
```
チャンネル1の音色はチャンネル2と同じ27番の音色になる

```
CALL VOICE(@20,@10)
CALL VOICE(,, @21)
```
チャンネル1の音色はチャンネル3と同じ21番の音色になる

```
CALL VOICE(A,, B)
```
チャンネル1の音色はチャンネル3と同じBという配列変数で設定される音色になる

音色番号	音色名	略号	OPLL VOICE	音色番号	音色名	略号	OPLL VOICE
0	Piano 1	Piano 1	3 ピアノ	32	Piano 3	Piano 3	
1	Piano 2	Piano 2		33	Electric Piano 2	Electpia2	14 ウッドベース
2	Violin	Violin	1 バイオリン	34	Santool 2	Santool2	
3	Flute 1	Fulute	4 フルート	35	Brass	Brass	
4	Clarinet	Clarinet	5 クラリネット	36	Flute 2	Flute2	
5	Oboe	Oboe	6 オーボエ	37	Clavicode 2	Clavicd2	
6	Trumpet	Trumpet	7 トランペット	38	Clavicode 3	Clavicd3	
7	Pipe Organ	PipeOrgan		39	Koto 2	Koto 2	
8	Xylophone	Xylophone		40	Pipe Organ 2	PipeOrg2	
9	Organ	Organ	8 オルガン	41	PohdsPLA	PohdsPLA	
10	Guitar	Guitar	2 ギター	42	RohdsPRA	RohdsPRA	
11	Santool 1	Santool		43	Orch L	Orch L	
12	Electric Piano 1	Elecpian	15 エレキベース	44	Orch R	Orch R	
13	Clavicode 1	Clavicod		45	Synthesizer Violin	SynViol	
14	Harpsicode 1	Harpsicd	11 ハープシコード	46	Synthesizer Organ	SynOrgan	
15	Harpsicode 2	Harpscd2		47	Synthesizer Brass	SynBrass	
16	Vibraphone	Vibraphn	12 ビブラフォン	48	Tube	Tube	9 ホルン
17	Koto 1	Koto		49	Shamisen	Shamisen	
18	Taiko	Taiko		50	Magical	Magical	
19	Engine 1	Engine		51	Huwawa	Huwawa	
20	UFO	UFO		52	Wander Flat	WnderFlt	
21	Synthesizer bell	SynBell		53	Hardrock	Hardrock	
22	Chime	Chime		54	Machine	Machine	
23	Synthesizer bass	SynBass	13 シンセベース	55	Machine V	MachineV	
24	Synthesizer	Synthsiz	10 シンセ	56	Comic	Comic	
25	Synthesizer Percussion	SynPercu		57	SE-Comic	SE-Comic	
26	Synthesizer Rhythm	SynRhyth		58	SE-Laser	SE-Laser	
27	Harm Drum	Harp Drum		59	SE-Noise	SE-Noise	
28	Cowbell	Cowbell		60	SE-N Star 1	SE-Star	
29	Close Hi-hat	ClseHiht		61	SE-Star 2	SE-Star2	
30	Snare Drum	SnareDrum		62	Engine 2	Engine 2	
31	Bass Drum	BassDrum		63	Silence	Silence	

表 4.4　音色ライブラリー覧表

リスト 4.8　VOICE 命令による音色の設定

```
100 CALL MUSIC
110 DIM A%(15)
120 FOR I=0 TO 15
130 READ A$:A%(I)=VAL("&H"+A$)
140 NEXT I
150 CALL VOICE(A%,A%,A%)
160 PLAY #2,"O4L4V15C","O4L4V15E","O4L4V15G"
170 END
180 DATA 6C43,6573,6948,7468,1800,8,0,0,8A3,76F1,80,0,82,17F2,80,0
```

VOICE COPY

●書式●　CALL VOICE COPY(〈パラメータ 1〉, 〈パラメータ 2〉)

配列と音色ライブラリ(0～63)の間でのデータの転送を行います.

パラメータ1の音色パラメータをパラメータ2に転送します. @と数式が指定されたときは, その数式の結果で指定される音色番号の音色データが対象となります.

ソース(パラメータ 1)に指定できる音色番号は0～63のうちOPLL VOICEの欄に指定がある音色の番号です.

デスティネーション(パラメータ 2)に指定できる音色番号は63に限ります.

ソース(パラメータ 1)にOPLL VOICEの欄に指定のない音色の番号を指定すると "Illegal function call" となります. @記号がない場合の変数名は配列変数とみなされ, その内容が転送の対象になります.

音色パラメータの内容は, 表4.5のとおりです. 1つの音色パラメータは32バイトの長さがあります. 配列変数にコピーする場合には, あらかじめDIM文によって32バイト分の配列の長さを宣言しなければなりません. たとえば, 整数型ならば,

```
DIM A%(15)
```

のように宣言します.

表 4.5 にならって音色パラメータの内容を配列変数に書き込み, VOICE 文で FM 音源のチャンネルに直接書き込むか, または VOICE COPY 文で音色番号@63 に書き込めば, オリジナルな音色を使うことができます.

オフセット	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ヘッダ								
0〜7	音色名(8文字)							
8〜9	ボイス移調							
10	AMD*	PMD*	AMD/PMD*	固定ピッチ*	フィードバック			アルゴリズム*
11〜15	(予　約)							
オペレータ0								
16	AM	PM	EG	KSR	MULT			
17	レベルキースケール		トータルレベル					
18	アタックレイト				ディケイレイト			
19	サスティンレベル				リリースレイト			
20	ベロシティセンシビリティ(0〜8)*							
21〜23	(予　約)							
オペレータ1								
24	AM	PM	EG	KSR	MULT			
25	レベルキースケール		トータルレベル*					
26	アタックレイト				ディケイレイト			
27	サスティンレベル				リリースレイト			
28	ベロシティセンシビリティ(0〜8)*							
29〜31	(予　約)							

＊MSX-AUDIO との互換性を保つために用意されているもので, MSX-MUSIC では無視される

表 4.5　音色パラメータ表

リスト 4.9 VOICE COPY 命令による音色の設定

```
100 CALL MUSIC
110 DIM A%(15)
120 FOR I=0 TO 15
130 READ A$:A%(I)=VAL("&H"+A$)
140 NEXT I
150 CALL VOICE COPY (A%,@63)
160 PLAY #2,"@6304L4V15C","@6304L4V15E","@6304L4V15G"
170 END
180 DATA 6C43,6573,6948,7468,1800,8,0,0,8A3,76F1,80,0,82,17F2,80,0
```

MSX－AUDIO のステートメント

参考として，MSX-AUDIO 拡張 BASIC のステートメントで，MSX-MUSIC では使えないステートメントをあげておきます．これらを指定するとすべて〝Illegal function call〟となります．

・ADPCM／PCM 関係のステートメント
CALL CONVA，CALL CONVP，CALL COPY PCM
CALL LOAD PCM，CALL PCM FREQ，CALL PCM VOL
CALL PLAY PCM，CALL REC PCM，CALL SAVE PCM
CALL SET PCM

・インスツルメント関係のステートメント
CALL INMK，CALL KEY ON／KEY OFF，CALL MK PCM
CALL MK TEMPO，CALL MK VEL，CALL MK VOICE
CALL MK VOL

・MK 記録関係のステートメント
CALL APPEND MK，CALL CONT MK，CALL MK STAT
CALL PLAY MK，CALL REC MK，CALL RECMOD

・外部プログラムの呼び出し
CALL SYNTHE，CALL APEEK，CALL APOKE

3 ミュージック機能パワフル活用

OPLL YM2413の機能

MSX-MUSIC拡張BASICでは，OPLLに直接値を書き込みOPLLを制御することができるようになっています．ここではOPLLの機能を直接使うことができるように，OPLLのレジスタを解説していきます．レジスタ番号は16進数で表されています．

■ レジスタ0，1

レジスタ0はモジュレータオペレータへの，レジスタ1はキャリアオペレータへの設定になります．

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
00	AM	VIB	EGTYP	KSR	MULTI			
01	AM	VIB	EGTYP	KSR	MULTI			

表4.6　レジスタ0，1

ビット3〜0(MULTIPLE)

オペレータ(モジュレータ，キャリア)の周波数を制御します．後述のBLOCK／F-Numberで設定された周波数とMULTIに設定した数値を掛けたものをそのオペレータの周波数とします．ただし，MULTIが0だった場合は，1／2を掛けます．

ビット4(KSR)

自然楽器，とくに打弦楽器では，同じ楽器であっても，音程によって微妙に音の立ち上がり，立ち下がりのスピードが違っています．KSRは，この変

化をシュミレートするものです．高音部では変化が速くなり，低音部では変化が遅くなるように，エンベロープの変化の速さを補正しています．0のときは比較的小さい補正量で，1のときは大きく補正します．

ビット5(EG-TYP)

エンベロープのパターンが，持続音タイプか減衰音タイプかの切り替えをします．0のときは，減衰音タイプのエンベロープパターンが選択され，1のときは持続音タイプのエンベロープパターンが選択されます．

ビット6(VIB)

ビブラートのON／OFFスイッチです．このビットを1にすると，そのオペレータにはビブラートがかかります．ビブラートとは，音程を周期的に微妙に変化させ，音がゆれている感じを表現することができます．

ビット7(AM)

振幅変調のON／OFFスイッチです．このビットを1にすると，そのオペレータには振幅変調がかかります．モジュレータ側のオペレータにAMをかけると音色が周期的に変化するようになります．キャリア側のオペレータにAMをかけると音量が周期的に変化するようになります．

■ レジスタ2，3

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
02	KSL		Total Level					
03	KSL		~~DM~~	DC	DM	FB		

表4.7　レジスタ2，3

ビット5〜0(Total Level：レジスタ2)

レジスタ2の，ビット5〜0は，モジュレータオペレータの出力の大きさを決めます．この値は最大値に対する減衰量を表し，0〜63までの範囲で指定します．0のときが一番出力が大きく，63のときが一番出力が小さくなります．

値が小さいほどキャリアの出力が大きく変化し，明るい音になります．

ビット7〜6(KSL)

音程によって音の立ち上がり，立ち下がりの速さが変化することは，KSRのところで説明しましたが，同じように，音色も音程によって変化します．このレジスタでは，この変化をシュミレートします．

音程が上がると，オペレータの出力を絞って，音量や音色を変化させます．1〜3までの範囲で設定します．値を大きくすると，音程に対する減衰量が大きくなります．

レジスタ2のKSLはモジュレータオペレータへの，レジスタ3のKSLはキャリアオペレータへの設定になります．

ビット5(DM：レジスタ3)

モジュレータの出力を半波整流します．

ビット4(DC：レジスタ3)

キャリアの出力を半波整流します．

ビット2〜0(FEEDBACK：レジスタ3)

モジュレータの出力を自分自身にフィードバックします．0のときはフィードバックしません．1〜7の範囲で設定します．数値を大きくしていくと，鋭い音色になっていきます．

■ レジスタ4，5

レジスタ4はモジュレータオペレータへの，レジスタ5はキャリアオペレータへの設定になります．

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
04	AR				DR			
05	AR				DR			

表4.8　レジスタ4，5

ビット7〜4(ATTACK RATE)

アタックレイトは，音の立ち上がりの時間を指定します．値は0から15までで，値が大きくなるほど，立ち上がりの時間は短くなります．

ビット3〜0(DECAY RATE)

ディケイレイトは，音量が最大になってからの，音の減衰時間を指定します．値は0から15までで，値が大きくなるほど減衰する時間は短くなります．

■ レジスタ6，7

レジスタ6はモジュレータオペレータへの，レジスタ7はキャリアオペレータへの設定になります．

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
06	SL				RR			
07	SL				RR			

表4.9　レジスタ6，7

ビット7〜4(SUSTAIN LEVEL)

サスティンレベルは，エンベロープパターンが減衰音タイプの場合は，ディケイモードでの減衰が，このレベルになると，その後はリリースモードに変わるという変化点を示し，持続音タイプのときは，ディケイモードでの減衰がこのレベルに達すると，その後はこのレベルを保持するという変化点を示します．値の範囲は0〜15で，最大値に対する減衰量で指定します．ですから，値が大きくなるほど音量は小さくなります．

ビット3〜0(RELEASE RATE)

リリースレイトは，エンベロープパターンが減衰音タイプの場合は，サスティンレベル前の減衰量をディケイレイトで表し，サスティンレベル後の減衰量をリリースレイトで表します．持続音タイプのときは，KEY OFFからの音の減衰量を示します．

■ レジスタ 10～18 と 20～28

レジスタ 10 から順番にチャンネル 1 から 9 に対応しています．同様にレジスタ 20 から順番にチャンネル 1 から 9 に対応しています．

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
10 ↓ 18	F-Number(ビット7～ビット0)							

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
20 ↓ 28			SUS ON／OFF	KEY ON／OFF	BLOCK			F-Numder ビット8

表 4.10　レジスタ 10～18 と 20～28

レジスタ 10～18　ビット 7～0　レジスタ 20～28　ビット 0(F-Number)

F-Number は，レジスタ 10～18 の 8 ビットとレジスタ 20～28 の下位 1 ビットの合計 9 ビットで表します．F-Number は音階を与えるデータで，FM 音源の基準クロックを対して，目的とする周波数が基準クロックのなん倍であるかという情報で、目的の周波数を表現しています．しかし，そのままでは，ビット数が多くて管理が複雑になるため後述のオクターブを表す 3 ビットの BLOCK と 9 ビットの F-Number で表しています．

目的の周波数から，F-Number と BLOCK を求めるには，次の式を使います．

BLOCK＝オクターブデータ
Fmus＝発生したい周波数(Hz)
F-Number＝(Fmus×2^18÷50000)÷2^(BLOCK-1)

例，A4＝440Hz の場合

BLOCK＝4
F-Number＝(440×2^18÷50000)÷2^(4-1)＝288

レジスタ 20〜28　ビット 3〜1(BLOCK)

BLOCKはオクターブ情報を与えます.

ビット 4(KEY-ON/OFF)

鍵盤のON/OFFに相当するビットです. このビットを"1"にするとそのチャンネルがONとなり発音します. "0"にするとKEY-OFFです.

ビット 5(SUS-ON/OFF)

このビットを1にするとKEY-OFFからのRR(リリースレイト)が5になります.

■ レジスタ 0E

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0E			RHYTHM	BD	SD	TOM	TOP-CY	HH

表 4.11　レジスタ 0E

ビット 5(RHYTHM)

リズムモードの選択と, 各リズム楽器のON/OFFをコントロールします. ビット5が1のとき, OPLLはリズム音モードになります. 7〜9までのチャンネルはリズム音発声用のチャンネルとなります. したがってメロディー音は6チャンネルに制限されます.

ビット4〜0は各リズム音源のON／OFFを制御します. このときは, レジスタ26, 27, 28のKEY-ONビットはつねに0にしておきます.

各リズム音の記号と音色の対応を以下に示します.

記号	音色名
BD	バスドラム
SD	スネアドラム
TOM	タム
T-CYM	トップシンバル
HH	ハイハット

■ レジスタ 30～38

レジスタ 30 から順番にチャンネル 1 から 9 に対応しています.

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
30 ↓ 38	INST				VOL			

表 4.12　レジスタ 30～38

ビット 7～4(INST)

OPLL の音色番号を設定します. 音色番号は 0～15 の範囲で設定します. 音色番号が, 0 のときは, レジスタ 0 から 7 で設定した音色になります.

音色番号	音 色 名	音色番号	音 色 名
0	オリジナル音色	8	オルガン
1	バイオリン	9	ホルン
2	ギター	10	シンセ
3	ピアノ	11	ハープシコード
4	フルート	12	ビブラフォン
5	クラリネット	13	シンセベース
6	オーボエ	14	ウッドベース
7	トランペット	15	エレキベース

表 4.13　OPLL VOICE 一覧表

ビット 3～0(VOL)

各チャンネルの音量を指定します. 0～15 の範囲で指定しますが, この値は最大値に対する減衰量なので, 0 を指定した場合がいちばん音が大きく, 15 を指定したときが一番音が小さくなります.

リズムモードの場合は，レジスタ 36, 37, 38 で各リズムの音量を表 4.14 のように指定します．値の指定の仕方は同様です．

レジスタ	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
36					BD			
37	HH				SD			
38	TOM				T-CYM			

表 4.14　レジスタ 36，37，38

活用サンプルプログラム①
FM音色エディタ

MSX-MUSICに使われている，OPLLは15個の音色を内蔵していて，これだけでも，PSGに比べれば充分優れているといえるのですが，さらに1音色のオリジナルな音色を作ることができます．これは実際には，OPLLのオリジナル音色レジスタに，パラメータを設定する事によって，行われています．OPLLにあらかじめ設定されていない音色(表4.4参照)は，すべてこのオリジナル音色レジスタへ値を書き込むことによって実現されています．

ここでは，自分で音色を作るための音色エディタを紹介します．

■ 音色エディタの操作

このプログラムでは，キャリアオペレータのパラメータ変更画面と，モジュレータオペレータのパラメータ変更画面とに分かれています．実行した直後は，モジュレータオペレータのパラメータ変更画面になっています．(巻頭口絵参照)

画面右上にCARRIER，MODULATORの文字が表示され，現在どちらのオペレータの値を変更しているのかを示します．

ESCキーを押すとCARRIER画面，MODULATOR画面の2つを切り替えることができます．また，INSキーを押すとPLAYデータ画面に切り替わります．PLAYデータ設定画面については，あとで説明します．

画面上にある，手の形をしたアイコンがカーソルです．これは，カーソルキーを押すことによってそれぞれのパラメータの上に移動します．

アタックレイト(AR)，ディケイレイト(DR)，サスティンレベル(SL)，リリースレイト(RR)，(モジュレータならば，加えてフィードバックレベル(FBL)とトータルレベル(TL))の値は，画面に描かれたスライダースイッチをカーソルで指定して，それを上下することによって設定します．AR，DR，RRはそれぞれ，上側にするほど，オペレータの出力の上がり下がりにかかる時間が長くなります．SLは，上に上げるほど，音量が大きくなります．

カーソルキーで変更するスライダーの位置へカーソルを移動し，スペースキーを押します．そのままで，カーソルキーの上下キーを押すことによって，

スライダーを上下することができます．スペースキーを離せば，決定されます．

MULTI，EG-TYP，KSL，KSRは，カーソルで選択して，スペースキーを押すと1つずつ値が大きくなっていきます．最大値の次は，最小値になります．

AM，VIB，DISも同様に，カーソルで選択し，スペースキーを押すことによって，ON／OFFが交互に切り替わります．

Pキーを押すと，後で説明するPLAYデータ設定画面で設定した音程，長さの音を鳴らして，音色をテストすることができます．

画面上側は，エンベロープパターンの表示ウィンドウです．エンベロープパターンは，キーオン(キーを押したとき)からキーオフ(キーを離したとき)までの時間や，KSL，KSR等のパラメータの設定で，かなり違ってきます．このプログラムでのエンベロープパターンの表示は，キーオンからキーオフまでの時間とKSL，KSRが，完全に無視されていますから，そのことを差し引いて考えなければなりません．また，エンベロープパターンの線の傾きや，サスティンレベルの時間は，実際の変化の割合とは大幅に違っています．このエンベロープパターンの表示は，単に目安として考えてください．

PLAYデータ設定画面では，音色をテストするための音程や，音の長さ，セーブ，ロード等を設定することができます．カーソルキーの上下キーで，機能を選択し，スペースキーで決定します．それぞれ，メッセージに従って値を入力し，リターンキーで決定します．INSキーかESCキーを押せば，パラメータ変更画面に戻ります．

PLAY DATAを選択すると，Pキーを押したときに出る音の音程を設定することができます．音程は，A～Gで設定します．プログラム中では，単に設定されている文字列をPLAY文に代入して実行しているだけですから，〝CDEFG〟とすれば，ドレミファソと演奏することもできます．ただし，文字列の長さには限界があり，64文字までしか入力することができません．また，MMLを指定すれば，TnやOn，Vnも設定できます．たとえば，〝T100L1O4V15A〟という設定も可能です．

〝：〟で区切って〝A：B：C〟とすれば，和音を鳴らすこともできます．ここではリズム音なしのモードを使っているので，最大9和音まで出せます．

PLAYデータはデフォルトでは,

"T120L4O4V15A"

と設定されています.

PLAY LENGTHを選択すると,音の長さを設定できます.データは1から64まで指定できます.この設定では,上記のPLAY DATA設定で設定されたMML文字列の先頭に自動的に,"Ln"を付加します.デフォルトでは"L4"に設定されています.ただし,PLAY DATA設定のMML文字列の中に"Ln"を設定している場合は,そちらが優先されます.

SAVE, LOADを選択すると,作ったデータをセーブ,ロードすることができます.選択すると,ファイルネームを聞いてきます.ファイルネームを入力します.

セーブでは,音色パラメータを,

```
10000 DATA 6C43,6573,6948,7468,1800,8,0,0,8A3,76F1,80,0,82,
17F2,80,0
```

のようなプログラムの形でアスキーセーブしています.実際にプログラムに組み込むときは,プログラムとセーブされたデータをマージして,次のようにして使ってください.

```
100 CALL MUSIC
110 DIM A%(15)
120 FOR I=0 TO 15
130 READ A$:A%(I)=VAL("&H"+A$)
140 NEXT I
150 CALL VOICE COPY (A%,@63)
160 PLAY #2,"@63CDEFG"
170 END
180  DATA  6C43,6573,6948,7468,1800,8,0,0,8A3,76F1,80,0,82,
17F2,80,0
```

VOICE COPYを選択すると，MSX-MUSICに内蔵されている音のデータを読み込んで，そのパラメータをエディットすることができます．音色は表4.4の音色番号で指定します．ただし，OPLLに内蔵されている音色を指定することはできません．

このエディタを終了するときは，CTRL+STOP(CTRLキーを押しながらSTOPキーを押す)を押すと終了します．

エディタを終了した時点では，作った音色のパラメータは@63に格納されたままになっています．CALL MUSIC文を実行しても，@63に設定されているパラメータは初期化されませんから，1度エディタを終了してしまっても，もう1度実行すれば，また，それまでエディットしていた音色を変更することができます．

このエディタでエディットしたあとに，次に紹介するFM音キーボードを実行し，音色に@63番を選べば，そのままFM音源キーボード上でも使うことができます．

■ 実際の音色の作り方

FM音源での音の作り方はとても難しいものです．実際に作りたい音の，波形，倍音成分の含まれ方，時間的変化の仕方を考えながらパラメータを設定していかなければなりませんが，これには，相当の鍛錬が必要です．

最初は，内蔵されている音色のパラメータを読み込んで，そのパラメータを少しずつ変化させてみることから始めるといいでしょう．しかし，やみくもにパラメータを変化させても，目的の音色は得られないでしょうから，ここでは，簡単にどのパラメータが音色にどのように影響するのかを簡単に説明しておきます．

・MULTIPLE

MULTIPLE(画面では，MULTI)は，出したい音の周波数の，何倍の周波数を実際にオペレータに加えるかを指定します．たとえば，4オクターブのAの音ならば周波数は440Hzですから，MULTIPLEを2にすれば，2倍の880Hzの周波数がオペレータに入力されます．ただし，0の時は，1/2倍になります．

モジュレータ側の周波数を高くしていくと，倍音成分の含まれ方が，基本の周波数に近い倍音成分が多く，基本の周波数から離れた倍音成分が多くなっていきます．キャリア側の周波数を高くしていくと，ピッチが高くなっていきます．通常 MULTIPLE は，モジュレータ側は，1 か 2，キャリア側は，1 を設定すればよいでしょう．

・フィードバックレベル

フィードバックレベル(FBL)は，モジュレータの出力をモジュレータ自身にフィードバックする量を設定します．フィードバックレベルを大きくしていくと，だんだん鋭い音色になっていきます．そして最大にすると，ホワイトノイズを発声するようになります．このパラメータは，トランペットのような金管の楽器の音を作る時に使用するとよいでしょう．

・オペレータ出力レベル

オペレータの出力レベルを変化させます．これは，エンベロープパターンを変化させることによって作ります．また，次で説明する KSL，KSR も影響します．さらに，モジュレータでは，トータルレベル(TL)も出力レベルを左右します．

エンベロープパターンは，アタックレイト(AR)，ディケイレイト(DR)，サスティンレベル(SL)，リリースレイト(RR)，EG-TYPE(減衰音と持続音の種別)によって表現します．

モジュレータ側の出力レベルを変化させることによって，音色の変化をつけることができます．出力レベルを上げていくと，丸い音色から明るい音色へと変化していきます．また，エンベロープパターンは，含まれている倍音成分の音量の時間的変化を作ります．

キャリアの出力レベルの変化は音量の変化を作ります．たとえばピアノのような打弦楽器の場合には，音の立ち上がりは早く変化するように設定し，時間がたつにつれて少しずつ音量が小さくなっていくようにします．トランペットのような金管楽器ならば，音の立ち上がりは比較的緩やかにして，音量は持続するように設定します．

・KSR，KSL

レイトのキースケール(KSR)は，たとえば，打弦楽器では，同じ楽器であっても，音程によって微妙に音の立ち上がり，立ち下がりのスピードが違ってくることをシュミレートします．また，レベルのキースケール(KSL)では，音程が高くなるほど，音量が小さくなっていくことをシュミレートします．

実際には，音程が上がると，オペレータの出力を絞って，音量や音色を変化させます．値を大きくすると，音程に対する音量の小さくなる割合が大きくなります．

・VIB，AM，DIS

ビブラート(VIB)のON／OFFや振幅変調(AM)のON／OFF，半波整流(DIS)のON／OFFを決めます．VIBをONにすると，ピッチが周期的に微妙に変化し，音がゆれている感じを出すことができます．

モジュレータ側のオペレータのAMをONにすると，音色が周期的に変化するようになり，キャリア側のオペレータのAMをONにすると，音量が周期的に変化するようになります．

それぞれのパラメータの意味は，だいたい上記のようになります．しかし，FM音源では，それぞれのパラメータの変化が，直線的に結果となって表れないので，実際に何度もやってみて，どのパラメータを変化させるとどのように音が変化していくか，体得していかなければならないでしょう．

リスト 4.10 FM 音色エディタ FMEDIT.BAS

```
1000 'MSX-MUSIC TONE COLOR EDITOR FOR YM2413(OPLL)
1010 CALL MUSIC(0,0,1,1,1,1,1,1,1,1,1)
1020 CLEAR 2000:MAXFILES=2
1030 SCREEN 5,2:OPEN "GRP:" FOR OUTPUT AS #1
1040 DIM PX(6),PY(6),PM(6,6,1),MA(6,6),R1(15),R2(15),R3(15),D$(8),D1$(
8),D2$(8),T%(15),OX(2),OY(2)
1050 DEFFNT=VARPTR(T%(0))
1060 CL$=STRING$(3,CHR$(8))+CHR$(15)+STRING$(3,CHR$(8))+STRING$(9,CHR$
(0))
1070 RESTORE 1200
1080 FOR I=0 TO 15:READ R1(I):NEXT I
1090 FOR I=0 TO 15:READ D:R2(I)=-D:NEXT I
1100 FOR I=0 TO 15:READ R3(I):NEXT I
1110 FOR I=0 TO 6:READ MA(I,0):NEXT I
1120 FOR I=1 TO 6:READ MA(2,I):NEXT I
1130 FOR I=0 TO 6:READ PX(I):NEXT I
1140 FOR I=0 TO 6:READ PY(I):NEXT I
1150 FOR I=0 TO 8:D1$(I)="T120@63L404V15":NEXT I
1160 D$(0)="A":GOSUB 2760
1170 GOSUB 3350:CALL VOICE COPY(@1,T%):GOSUB 2790
1180 PO=0:OX(1)=2:MD=0:CL=1:RM=1:RC=1:GOSUB 1610
1190 GOTO 1280
1200 'DATA
1210 DATA 0,40,35,30,26,22,18,15,12,9,7,5,3,2,1,0
1220 DATA 0,.8,.9,1,1.2,1.4,1.7,2,2.5,3.3,4.3,6,10,15,30,60
1230 DATA 60,56,52,48,44,40,36,32,28,24,20,16,12,8,4,0
1240 DATA 63,15,15,15,15,15,15
1250 DATA 1,3,1,1,1,1
1260 DATA 24,48,136,160,184,208,232
1270 DATA 124,132,140,148,172,180,188
1280 'MAIN
1290 T=STICK(PO)
1300 X=X-(T=3 AND MD<>2 AND X<6)+(T=7 AND MD<>2 AND X>(2*MD))
1310 Y=Y-(T=5 AND(X=2 OR MD=2)AND Y<6+(MD=2))+(T=1 AND(X=2 OR MD=2)AND
 Y>0)
1320 IF OX(MD)<>X OR OY(MD)<>Y THEN GOSUB 2520:OX(MD)=X:OY(MD)=Y:GOTO
1290
1330 K$=INKEY$
1340 IF K$=CHR$(&H1B) THEN GOSUB 1590:GOTO 1380
1350 IF K$=CHR$(&H12) THEN GOSUB 1560:GOTO 1380
1360 IF (K$="P" OR K$="p") AND MD<>2 THEN GOSUB 2720
1370 IF STRIG(PO) THEN 1390
```

```
1380 GOTO 1290
1390 'IF STRIG(PO)=-1
1400 PT=1:GOSUB 2520
1410 IF MD=2 THEN 1740
1420 IF X=2 THEN 1480
1430 T=STICK(PO)
1440 D=PM(X,0,MD)-(T=5 AND PM(X,0,MD)<MA(X,0))+(T=1 AND PM(X,0,MD)>0)
1450 IF D<>PM(X,0,MD) THEN PM(X,0,MD)=D:GOSUB 2520:GOSUB 2610
1460 IF STRIG(PO)=0 THEN 1520
1470 GOTO 1430
1480 'CHANGE BY SPACE KEY
1490 PM(2,Y,MD)=PM(2,Y,MD)+1
1500 IF PM(2,Y,MD)>MA(2,Y) THEN PM(2,Y,MD)=0
1510 GOSUB 2670
1520 IF MD<>2 AND((X=2 AND Y=1) OR X>=3) THEN GOSUB 3100
1530 GOSUB 2970
1540 PT=0:GOSUB 2520
1550 GOTO 1290
1560 'MODE CHANGE
1570 IF MD=2 THEN MD=MO:GOTO 1610
1580 MO=MD:MD=2:GOTO 1610
1590 '
1600 MD=MD+1:IF MD>=2 THEN MD=0
1610 'MODE CHANGE2
1620 SET PAGE ,MD
1630 IF MD=2 THEN 1660
1640 IF MD=0 AND RM<>0 THEN RM=0:GOSUB 1690:GOTO 1660
1650 IF MD=1 AND RC<>0 THEN RC=0:GOSUB 1690
1660 X=OX(MD):Y=OY(MD):GOSUB 2520
1670 SET PAGE MD,MD
1680 RETURN
1690 '
1700 FOR X=MD*2 TO 6:GOSUB 2610:NEXT X
1710 FOR Y=0 TO 6:GOSUB 2670:NEXT Y
1720 GOSUB 3100
1730 RETURN
1740 'MD=2
1750 ON Y+1 GOTO 1760,1840,1920,1920,2150,2200
1760 'MML DATA INPUT
1770 I=0:KT=1:LN=64:PT$="INPUT MML":GOSUB 2330
1780 LN=INSTR(I$,":"):IF LN=0 THEN LN=LEN(I$)+1
1790 D$(I)=LEFT$(I$,LN-1)
1800 I$=MID$(I$,LN+1,LEN(I$))
```

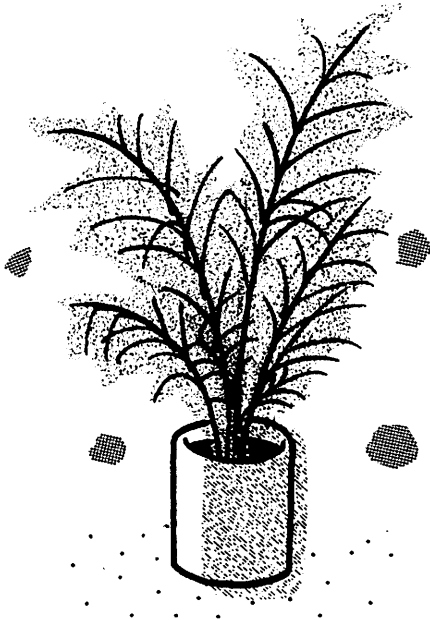
```
1810 IF I$<>"" THEN I=I+1:GOTO 1780
1820 GOSUB 2760
1830 GOTO 2300
1840 'INPUT LENGTH
1850 KT=2:LN=2:PT$="LENGTH (1=<n=<64)":GOSUB 2330
1860 IF VAL(I$)>64 OR VAL(I$)<1 THEN BEEP :GOTO 2300
1870 FOR I=0 TO 8
1880 D1$(I)="T120@63L"+I$+"04V15"
1890 NEXT I
1900 GOSUB 2760
1910 GOTO 2300
1920 'LOAD & SAVE
1930 LN=8:KT=0:PT$="INPUT FILENAME":GOSUB 2330
1940 IF I$="" THEN BEEP :GOTO 2300
1950 IF Y=3 THEN 2030
1960 'SAVE
1970 GOSUB 2790
1980 OPEN I$+".DAT" FOR OUTPUT AS #2
1990 PRINT #2,"10000 DATA ";
2000 FOR I=0 TO 14:PRINT #2, STR$(T%(I))+",";:NEXT I
2010 PRINT #2,STR$(T%(15))
2020 GOTO 2130
2030 'LOAD
2040 RM=1:RC=1
2050 OPEN I$+".DAT" FOR INPUT AS #2
2060 LINE INPUT #2,D$:D$=MID$(D$,11,LEN(D$)-11)
2070 FOR I=0 TO 14
2080 T%(I)=VAL(MID$(D$,1,INSTR(D$,",")-1))
2090 D$=MID$(D$,INSTR(D$,",")+1)
2100 NEXT I
2110 T%(15)=VAL(D$)
2120 GOSUB 2790
2130 CLOSE #2
2140 GOTO 2300
2150 'TRANSPOSE
2160 LN=6:KT=2:PT$="TRANSPOSE (-12799=<n=<12799)":GOSUB 2330
2170 IF VAL(I$)<-12799 OR VAL(I$)>12799 THEN BEEP :GOTO 2300
2180 TR=VAL(I$)
2190 GOTO 2300
2200 'VOICE COPY
2210 RESTORE 2290:LN=2:KT=2:PT$="INPUT TONE No.":GOSUB 2330
2220 IF VAL(I$)>64 THEN BEEP:GOTO 2300
2230 READ D$
```

```
2240 IF D$="END" THEN D=VAL(I$):GOTO 2270
2250 IF VAL(I$)=VAL(D$) THEN BEEP :GOTO 2300
2260 GOTO 2230
2270 CALL VOICE COPY(@D,T%):GOSUB 2790:RM=1:RC=1
2280 GOTO 2300
2290 DATA 0,2,3,4,5,6,9,10,12,14,16,23,24,33,48,END
2300 'RETURN MAIN
2310 LINE (0,80)-(255,200),4,BF
2320 GOTO 1520
2330 'KEY INPUT
2340 I$="":PRESET (8,85):PRINT #1,PT$:PRESET(8,100)
2350 K$=INKEY$:IF K$="" THEN 2350
2360 IF K$=CHR$(&HD) THEN RETURN
2370 IF K$=CHR$(&H8) THEN 2470
2380 IF KT=2 AND (K$="+" OR K$="-") THEN 2430
2390 IF K$>="0" AND K$=<"9" THEN 2430
2400 IF KT=1 AND K$=":" THEN 2430
2410 IF KT<>2 AND ((K$>="@" AND K$=<"Z") OR K$="_" OR K$=">" OR K$="<"
 OR (K$>="a" AND K$=<"z")) THEN 2430
2420 GOTO 2350
2430 'PRINT DATA
2440 PRESET (8+8*(LEN(I$)-(20*INT(LEN(I$)/20))),100+10*INT(LEN(I$)/20)
):PRINT #1,K$:I$=I$+K$
2450 IF LEN(I$)>LN THEN BEEP:GOTO 2470
2460 GOTO 2350
2470 'DELETE
2480 IF LEN(I$)=0 THEN 2350
2490 PRESET (8+8*((LEN(I$)-1)-(20*INT((LEN(I$)-1)/20))),100+10*INT((LE
N(I$)-1)/20)):PRINT #1," "
2500 I$=LEFT$(I$,LEN(I$)-1)
2510 GOTO 2350
2520 'CORSOR MOVE
2530 IF MD=2 THEN DX=80:DY=14+(10*Y):GOTO 2580
2540 IF X=0 THEN DY=113+PM(0,0,0)+4:GOTO 2570
2550 IF X=2 THEN DY=PY(Y):GOTO 2570
2560 DY=113+(PM(X,0,MD)*5)
2570 DX=PX(X)
2580 PUT SPRITE0,(DX,DY),1,PT*2
2590 PUT SPRITE1,(DX,DY),11,PT*2+1
2600 RETURN
2610 'DISPLAY SLIDER
2620 IF X=2 THEN RETURN
2630 IF X=0 THEN DY=113+PM(0,0,0) ELSE DY=113+(PM(X,0,MD)*5)
```

```
2640 IF MD=1 THEN SX=X-1 ELSE SX=X+2
2650 PUT SPRITE SX,(PX(X)-8,DY),,4:COLOR SPRITE$(SX)=CL$
2660 RETURN
2670 'DISPLAY NUMBER
2680 IF Y=<3 THEN D$=RIGHT$("  "+STR$(PM(2,Y,MD)),2):GOTO 2700
2690 IF PM(2,Y,MD)=1 THEN D$="ON " ELSE D$="OFF"
2700 PRESET (PX(2)-(8*2),PY(Y)-4):PRINT #1,D$
2710 RETURN
2720 'TEST TONE COLOR
2730 CALL VOICE COPY(T%,@63)
2740 PLAY #2,D2$(0),D2$(1),D2$(2),D2$(3),D2$(4),D1$(5),D2$(6),D2$(7),D
2$(8)
2750 RETURN
2760 'MML DATA SETUP
2770 FOR I=0 TO 8:D2$(I)=D1$(I)+D$(I):NEXT I
2780 RETURN
2790 'READ VOICE DATA
2800 TR=PEEK(FNT+8)*256+PEEK(FNT+9):IF TR>128 THEN TR=TR-256
2810 PM(1,0,0)=(PEEK(FNT+10) AND 14)/2
2820 FOR I=16 TO 24 STEP 8
2830 D=(I-16)/8
2840 PM(2,4,D)=-((PEEK(FNT+I) AND 128)<>0)
2850 PM(2,5,D)=-((PEEK(FNT+I) AND 64)<>0)
2860 PM(2,1,D)=-((PEEK(FNT+I) AND 32)<>0)
2870 PM(2,3,D)=-((PEEK(FNT+I) AND 16)<>0)
2880 PM(2,0,D)=(PEEK(FNT+I) AND 15)
2890 PM(2,2,D)=(PEEK(FNT+I+1) AND 192)/64
2900 PM(0,0,D)=(PEEK(FNT+I+1) AND 63)
2910 PM(3,0,D)=(PEEK(FNT+I+2) AND 240)/16
2920 PM(4,0,D)=(PEEK(FNT+I+2) AND 15)
2930 PM(5,0,D)=(PEEK(FNT+I+3) AND 240)/16
2940 PM(6,0,D)=(PEEK(FNT+I+3) AND 15)
2950 NEXT I
2960 RETURN
2970 'SET VOICE DATA
2980 IF TR<0 THEN D=TR+256 ELSE D=TR
2990 POKE FNT+8,D¥256
3000 POKE FNT+9,D MOD 256
3010 POKE FNT+10,PM(1,0,0)*2
3020 FOR I=16 TO 24 STEP 8
3030 D=(I-16)/8
3040 POKE FNT+I,PM(2,4,D)*128+PM(2,5,D)*64+PM(2,1,D)*32+PM(2,3,D)*16+P
M(2,0,D)
```

```
3050 POKE FNT+I+1,PM(2,2,D)*64+PM(0,0,D)
3060 POKE FNT+I+2,PM(3,0,D)*16+PM(4,0,D)
3070 POKE FNT+I+3,PM(5,0,D)*16+PM(6,0,D)
3080 NEXT I
3090 RETURN
3100 'PUT ENVELOP PATTERN
3110 LINE (8,8)-(247,88),15,BF
3120 EY=80:EX=16
3130 IF PM(3,0,MD)=0 THEN LINE (EX,EY)-STEP(200,0),CL:GOTO 3340
3140 LINE (EX,EY)-STEP(R1(PM(3,0,MD)),-60),CL
3150 IF PM(4,0,MD)=0 AND PM(5,0,MD)=0 THEN GOTO 3190
3160 IF PM(4,0,MD)=0 THEN LINE -STEP(80,0),CL:IF PM(2,1,MD)=0 THEN R1=
0:GOTO 3320:ELSE GOTO 3220
3170 LINE -STEP(INT((-60+R3(PM(5,0,MD)))/R2(PM(4,0,MD))),60-R3(PM(5,0,
MD))),CL
3180 IF PM(5,0,MD)=15 THEN 3340
3190 IF PM(2,1,MD)=0 THEN GOTO 3260
3200 'JIZOKUON
3210 LINE -STEP(40,0),8
3220 IF PM(6,0,MD)=0 THEN LINE -STEP(80,0),CL:GOTO 3340
3230 IF PM(4,0,MD)=0 THEN R2=0 ELSE R2=PM(5,0,MD)
3240 LINE -STEP(-R3(R2)/R2(PM(6,0,MD)),R3(R2)),CL
3250 GOTO 3340
3260 'GENNSUION
3270 IF PM(6,0,MD)=0 THEN R1=PM(5,0,MD):GOTO 3320
3280 IF PM(5,0,MD)=14 OR PM(5,0,MD)=13 THEN R1=15:GOTO 3300
3290 R1=13
3300 LINE -STEP(INT((-R3(PM(5,0,MD))+R3(R1))/R2(PM(6,0,MD))),R3(PM(5,0
,MD))-R3(R1)),8
3310 IF PM(5,0,MD)=14 THEN GOTO 3340
3320 R2=7
3330 LINE -STEP(INT((-R3(R1))/R2(R2)),R3(R1)),CL
3340 RETURN
3350 'INIT SCREEN
3360 SET PAGE 0,0:SCREEN 5,2:COLOR 15,4,4:CLS
3370 MD=0:MD$="MODULATOR":GOSUB 3480:GOSUB 3830
3380 SET PAGE 0,1:SCREEN ,2:COLOR 15,4,4:CLS
3390 MD=1:MD$="CARRIER":GOSUB 3480:GOSUB 3830
3400 SET PAGE 0,2:SCREEN ,2:COLOR 15,4,4:CLS:GOSUB 3830
3410 COLOR 15,7:RESTORE 3470
3420 FOR I=0 TO 5
3430 READ D$:PRESET (20,10+10*I):PRINT #1,D$
3440 NEXT I
```

```
3450 COLOR 15,4
3460 RETURN
3470 DATA "PLAY DATA  ","PLAY LENGTH","  SAVE   ","  LOAD   "," TR
ANSPOSE ","VOICE COPY "
3480 'INIT SCREEN SUB
3490 PRESET (8,0):PRINT #1,MD$
3500 LINE (8,8)-(248,100),15,BF
3510 IF MD=1 THEN GOTO 3560
3520 RESTORE 3800
3530 FOR I=1 TO 2
3540 READ Y,X,C$:PRESET (8*X,8*Y):PRINT #1,C$
3550 NEXT I
3560 RESTORE 3810
3570 FOR I=1 TO 11
3580 READ Y,X,C$:PRESET (8*X,8*Y):PRINT #1,C$
3590 NEXT I
3600 FOR I=0 TO 3
3610 LINE (160+(24*I),112)-(160+(24*I),200),1
3620 LINE (147+(24*I),117)-(147+(24*I),192),14
3630 FOR J=0 TO 15
3640 LINE(147+(24*I),117+(J*5))-STEP(4,0),14
3650 NEXT J
3660 NEXT I
3670 IF MD=1 THEN 3790
3680 FOR I=0 TO 1
3690 LINE (24+(24*I),112)-(24+(24*I),200),1
3700 NEXT I
3710 LINE (35,117)-(35,192),14
3720 FOR J=0 TO 15
3730 LINE (35,117+(J*5))-(39,117+(J*5)),14
3740 NEXT J
3750 LINE (11,117)-(11,180),14
3760 FOR J=0 TO 64 STEP 4
3770 LINE (11,116+J)-(15,116+J),14
3780 NEXT J
3790 RETURN
3800 DATA 13,2,TL,13,5,FB
3810 DATA 13,19,AR,13,22,DR,13,25,SL,13,28,RR
3820 DATA 15,9,MULTI=,16,8,EG-TYP=,17,11,KSL=,18,11,KSR=,21,12,AM=,22,
11,VIB=,23,11,DIS=
3830 'SET SPRITE DATA
3840 RESTORE 3920
3850 FOR I= 0 TO 4:S$=""
```

```
3860 FOR J= 1 TO 32
3870 READ D$:S$=S$+CHR$(VAL("&H"+D$))
3880 NEXT J
3890 SPRITE$(I)=S$
3900 NEXT I
3910 RETURN
3920 DATA E0,90,89,46,21,10,08,14,24,20,10,08,07,00,00,00
3930 DATA 00,60,90,48,04,04,04,04,02,01,01,02,84,48,30,00
3940 DATA 00,60,70,39,1E,0F,07,0B,1B,1F,0F,07,00,00,00,00
3950 DATA 00,00,60,B0,F8,F8,F8,F8,FC,FE,FE,FC,78,30,00,00
3960 DATA 00,00,F9,86,81,70,08,14,24,20,10,08,07,00,00,00
3970 DATA 00,60,90,48,04,04,04,04,02,01,01,02,84,48,30,00
3980 DATA 00,00,00,79,7E,0F,07,0B,1B,1F,0F,07,00,00,00,00
3990 DATA 00,00,60,B0,F8,F8,F8,F8,FC,FE,FE,FC,78,30,00,00
4000 DATA FF,FF,FF,FF,FF,FF,FF,00,00,00,00,00,00,00,00,00
4010 DATA FF,FF,FF,FF,FF,FF,FF,00,00,00,00,00,00,00,00,00
```

活用サンプルプログラム②
FM音源キーボード

MSX-MUSICは，PLAY文でいちいちプログラムを書かなければ，音楽を演奏させることはできません．そこで，MSXのキーボードを押すことによって，音楽を演奏することのできるプログラムを紹介します．

■ キーボードと音階の対応

このプログラムでは，キーボードを鍵盤に見たてて，キーを押すことによって演奏することができます．各キーは，ピアノの鍵盤と同じように設定されています．MSXのキーボードは，上側から4段の列に分かれていますが，このうちの上側の2段と下側の2段を，それぞれ1組として扱います．それぞれ上の段が黒鍵，下の段が白鍵に当たります．

実際のキーに対する音の割当は，次のようになっています．

下側の2段のうち，下の段は，Zキーがドの音に対応していて，そこから右側に向かってXキーから，レ，ミ，ファ，ソ，ラ，シ，と割り当てられています．その上の段はSキーがド#，Dキーがレ#のように半音が，割り当てられています．

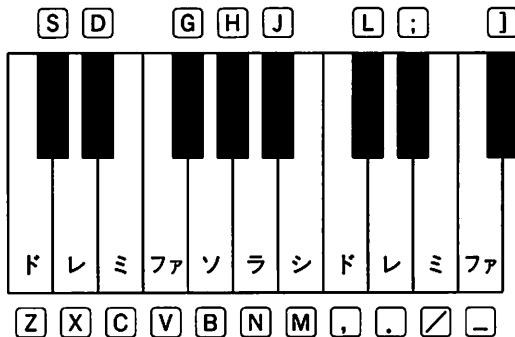



図4.10　キーボード下段の割り当て

上側の2段のうち，下の段はQキーがソの音に対応しています．Qキーから下の2段と同様に，ソ，ラ，シ，ド，レ，と割り当てられています．上の段は，2キーからソ#，3キーがラ#となって，下の2段と同様に音が割り当てられています．

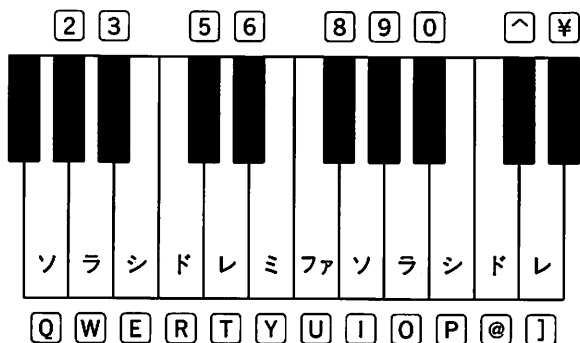



図4.11　キーボード上段の割り当て

音階は，下の段から上の段へと上がっていきます．全部で，3オクターブと2音分になります．つまり，1つのつながった鍵盤を2つに分けて上下に置いた形になっているわけです．

同時に6音まで発声できますから，複数のキーを押して，和音を弾くことができます．ただし，6個以上のキーが押された場合は，最後に押されたキーは無視されます．

■ 音色の切り替え

このプログラムは，上下のキーにそれぞれ別の音色を割り当てて演奏することもできます．F1キーを押すと，上下同じ音色のモードと，上下に別々の音色を設定する上下分割モードが切り替わります．このスイッチはトグルになっていますので，もう1度F1キーを押せば，上下同じ音色のモードに戻ります．

画面には，音色の名前とその番号が表示されています．また，画面の最下行には，現在セレクトされている音色が表示されます．ESC キーを押すと音色を選択するモードになります．カーソルキーでカーソルを移動して，リターンキーで決定します．

上下分割モードの時は，リターンキーで音色を決定したあと，上下どちらの段のキーに，その音を割り当てるかを設定します．カーソルキーの左で下のキーに，右で上のキーに割り当てられるので，どちらかを選択してリターンキーで決定します．ただし，名前の後ろに〝*〟マークの付いていない音色を同時に選択することはできません．

音色によっては，きちんと音階を発声できない音色もありますが，これは，バグではありません(とくにノイズを多く含んだ音や打楽器音など)．

■ プログラム中のマシン語

このプログラムでは，同時に複数のキーが押されたり,離されたりしたことを読み取るために，マシン語のプログラムで直接キーを読み込んでいます．

また，OPLL のレジスタに直接値を書き込むことによって音を発声させています．押されているキーに対応する音の周波数(F-number，BLOCK)をレジスタに書き込み，KEY-ON のビットをセットします．キーが離されたら，KEY-ON のビットをリセットします．

OPLL のレジスタに値を書き込むために，MSX-MUSIC の FM-BIOS のなかの WRTOPL ルーチンを使用しています．

リスト 4.11　FM 音源キーボード　FMKEYBD.BAS

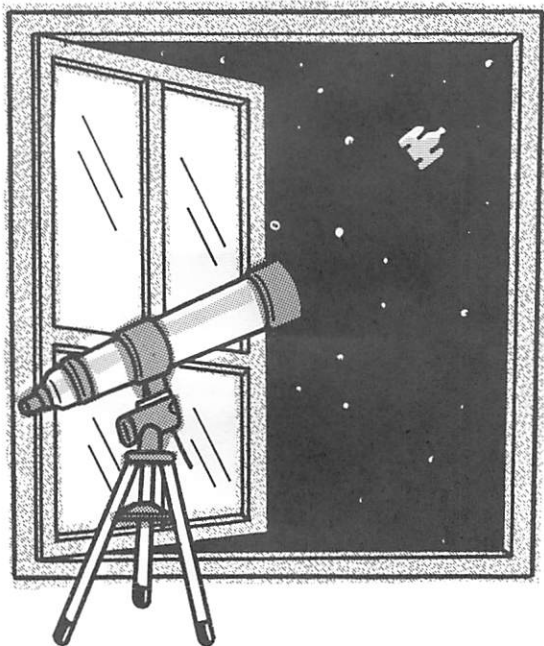
```
1ØØØ 'ｷｰﾎﾞｰﾄﾞ ｴﾝｿｳ PROGRAM
1Ø1Ø CLEAR 1ØØØ,&HBFFF
1Ø2Ø SCREEN Ø:WIDTH 4Ø:KEY OFF
1Ø3Ø CALL MUSIC
1Ø4Ø DIM TN$(63),T(1),OI(63)
1Ø5Ø GOSUB 2Ø5Ø
1Ø6Ø DEF USRØ=&HCØØØ:DEF USR1=&HCØØ3
1Ø7Ø DEF USR2=&HCØØ6:DEF USR3=&HCØØ9
1Ø8Ø DUMMY=USR1(DUMMY)
1Ø9Ø GOSUB 157Ø
11ØØ MD=Ø:T(Ø)=Ø:T(1)=2
111Ø GOSUB 145Ø
112Ø VL=Ø:D%=2*256+VL:DUMMY=USR3(D%)
113Ø 'MAIN
114Ø DUMMY=USRØ(DUMMY)
115Ø K$=INKEY$
116Ø IF K$=CHR$(&H1B) THEN GOSUB 119Ø
117Ø IF K$=CHR$(&H12) THEN GOSUB 14ØØ
118Ø GOTO 114Ø
119Ø 'ｵﾝｼｮｸ ﾍﾝｺｳ
12ØØ LOCATE X*13,Y+1:PRINT ">";
121Ø T=STICK(Ø)
122Ø X=X-(T=3 AND Y=<2Ø AND X<2)+(T=7 AND X>Ø)
123Ø Y=Y-((T=5 AND Y=2Ø AND X=Ø) OR (T=5 AND Y<2Ø))+(T=1 AND Y>Ø)
124Ø IF INKEY$=CHR$(&HD) THEN GOTO 127Ø
125Ø IF OX<>X OR OY<>Y THEN LOCATE OX*13,OY+1:PRINT " ";:OX=X:OY=Y:GOT
O 12ØØ
126Ø GOTO 121Ø
127Ø LOCATE X*13,Y+1:PRINT " ";
128Ø OS=Ø:S=Ø
129Ø IF MD=Ø THEN 137Ø
13ØØ LOCATE S*19+1,23:PRINT ">";
131Ø T=STICK(Ø)
132Ø S=S-(T=3 AND S=Ø)+(T=7 AND S=1)
133Ø IF INKEY$=CHR$(&HD) THEN GOTO 136Ø
134Ø IF OS<>S THEN LOCATE OS*19+1,23:PRINT " ";:OS=S:GOTO 13ØØ
135Ø GOTO 131Ø
136Ø IF OI(3*Y+X)=Ø AND OI(T(1-S))=Ø THEN BEEP:LOCATE S*19+1:PRINT " "
;:GOTO 12ØØ
137Ø T(S)=3*Y+X
138Ø GOSUB 145Ø
139Ø RETURN 114Ø
```

```
1400 'ﾓｰﾄﾞ ﾍﾝｺｳ
1410 DUMMY=USR2(DUMMY)
1420 IF MD=1 THEN MD=0 ELSE MD=1
1430 GOSUB 1450
1440 RETURN 1140
1450 'ｵﾝｼｮｸﾒｲ ﾉ ﾋｮｳｼﾞ ﾄ ｵﾝｼｮｸ ﾍﾝｺｳ
1460 LOCATE 0,23:PRINT STRING$(39," ");
1470 IF MD=0 THEN 1510
1480 LOCATE 2,23:PRINT "LowerKey "+TN$(T(0));
1490 LOCATE 21,23:PRINT "UpperKey "+TN$(T(1));
1500 GOTO 1520
1510 LOCATE 10,23:PRINT "Tone "+TN$(T(0));
1520 IF OI(T(1))=0 THEN CALL VOICE(@T(1))
1530 D%=1*256+OI(T(1)):DUMMY=USR3(D%)
1540 IF OI(T(0))=0 THEN CALL VOICE(@T(0))
1550 D%=OI(T(O)):DUMMY=USR3(D%)
1560 RETURN
1570 'ｶﾞﾒﾝ ﾉ ｼｮｷｶ
1580 RESTORE 1770
1590 FOR I=0 TO 63
1600 READ TN$(I)
1610 NEXT I
1620 LOCATE 5,0:PRINT "   MSX-MUSIC ORGAN PROGRAM"
1630 N=0
1640 FOR I=0 TO 20
1650 LOCATE 0,1+I
1660 FOR J=0 TO 2
1670 PRINT " ";RIGHT$("  "+STR$(N),2);" ";TN$(N);
1680 N=N+1
1690 NEXT J
1700 NEXT I
1710 LOCATE 0,22:PRINT " ";RIGHT$("  "+STR$(63),2);" ";TN$(63);
1720 RESTORE 2000
1730 FOR I=0 TO 63
1740 READ OI(I)
1750 NEXT I
1760 RETURN
1770 'ｵﾝｼｮｸﾒｲ ﾉ DATA
1780 DATA "Piano 1 *","Piano 2  ","Violin  *"
1790 DATA "Flute 1 *","Clarinet*","Oboe    *"
1800 DATA "Trumpet *","PipeOrgl ","Xylophon "
1810 DATA "Organ   *","Guiter  *","Santool  "
1820 DATA "ElecPian*","Clavicod ","Harpsicd*"
```

```
1830 DATA "Harpscd2 ","Vibraphn*","Koto     "
1840 DATA "Taiko    ","Engine   ","UFO      "
1850 DATA "SynBell  ","Chime    ","SynBass *"
1860 DATA "Synthsiz*","SynPercu ","SynRhyth "
1870 DATA "HarmDrum ","Cowbell  ","ClseHiht "
1880 DATA "SnareDrm ","BassDrum ","Piano 3  "
1890 DATA "ElecPia2*","Santool2 ","Brass    "
1900 DATA "Flute 2  ","Clavicd2 ","Clavicd3 "
1910 DATA "Koto 2   ","PipeOrg2 ","PohdsPLA "
1920 DATA "RohdsPRA ","Orch L   ","Orch R   "
1930 DATA "SynViol  ","SynOrgan ","SynBrass "
1940 DATA "Tube    *","Shamisen ","Magical  "
1950 DATA "Huwawa   ","WnderFlt ","Hardrock "
1960 DATA "Machine  ","MachineV ","Comic    "
1970 DATA "SE-Comic ","SE-Laser ","SE-Noise "
1980 DATA "SE-Star  ","SE-Star2 ","Engine2  "
1990 DATA "Orginal  "
2000 'ｵﾘｼﾞﾅﾙ ｵﾝｼｮｸ ﾄ ﾅｲｿﾞｳ ｵﾝｼｮｸ ｦ ｸﾍﾞﾂｽﾙ ﾃﾞｰﾀ
2010 DATA 3,0,1,4,5,6,7,0,0,8,2,0,15,0,11,0
2020 DATA 12,0,0,0,0,0,0,13,10,0,0,0,0,0,0,0
2030 DATA 0,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2040 DATA 9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2050 'ﾏｼﾝｺﾞ ｾｯﾄ
2060 RESTORE 2110
2070 I=0
2080 READ D$:IF D$="END" THEN 2100
2090 POKE &HC000+I,VAL("&H"+D$):I=I+1:GOTO 2080
2100 RETURN
2110 'ﾏｼﾝｺﾞ DATA
2120 DATA C3,0C,C0,C3,7A,C1,C3,E2,C1,C3,F2,C1,3E,00,32,1C
2130 DATA C2,3E,00,32,1D,C2,3A,1D,C2,CD,41,01,4F,3A,1C,C2
2140 DATA 47,04,CB,21,10,FC,30,04,16,00,18,02,16,FF,3A,1C
2150 DATA C2,47,3A,1D,C2,CB,27,CB,27,CB,27,80,4F,06,00,21
2160 DATA 27,C2,09,7E,FE,FF,28,16,4F,06,00,21,73,C2,09,7E
2170 DATA BA,28,0B,79,32,1F,C2,7A,32,1E,C2,CD,79,C0,3A,1C
2180 DATA C2,3C,E6,07,32,1C,C2,FE,00,20,AB,3A,1D,C2,3C,E6
2190 DATA 07,32,1D,C2,FE,00,20,9E,C9,3A,1E,C2,FE,00,20,05
2200 DATA CD,AF,C0,18,03,CD,89,C0,C9,21,21,C2,06,06,0E,00
2210 DATA 7E,FE,FF,28,05,23,0C,10,F7,C9,3A,1F,C2,77,5F,16
2220 DATA 00,21,73,C2,19,36,FF,79,32,20,C2,CD,D8,C0,C9,21
2230 DATA 21,C2,06,06,0E,00,3A,1F,C2,BE,28,05,23,0C,10,F9
2240 DATA C9,3E,FF,77,3A,1F,C2,5F,16,00,21,73,C2,19,36,00
2250 DATA 79,32,20,C2,CD,58,C1,C9,3A,1B,C2,FE,00,28,0C,3A
```

```
2260 DATA 1F,C2,FE,12,38,05,3A,18,C2,18,03,3A,19,C2,47,CB
2270 DATA 20,CB,20,CB,20,CB,20,3A,1A,C2,E6,0F,80,4F,CD,46
2280 DATA C1,3A,1F,C2,06,02,C6,05,FE,0C,38,05,D6,0C,04,18
2290 DATA F7,5F,16,00,21,67,C2,19,4E,78,CB,27,F6,31,47,CD
2300 DATA 23,C1,C9,3A,20,C2,C6,10,59,FD,2A,16,C2,DD,21,10
2310 DATA 41,CD,1C,00,3A,20,C2,C6,20,58,FD,2A,16,C2,DD,21
2320 DATA 10,41,CD,1C,00,C9,3A,20,C2,C6,30,59,FD,2A,16,C2
2330 DATA DD,21,10,41,CD,1C,00,C9,3A,1F,C2,06,02,C6,05,FE
2340 DATA 0C,38,05,D6,0C,04,18,F7,5F,16,00,21,67,C2,19,4E
2350 DATA 78,CB,27,F6,21,47,CD,23,C1,C9,06,30,21,73,C2,AF
2360 DATA 77,23,10,FC,CD,88,C1,C9,06,04,C5,3E,04,90,4F,21
2370 DATA C1,FC,5F,16,00,19,7E,87,30,19,06,04,C5,3E,24,90
2380 DATA 07,07,B1,CD,BB,C1,C1,28,10,10,F1,AF,32,17,C2,C1
2390 DATA 10,D8,C9,79,CD,BB,C1,20,F2,C1,C9,32,17,C2,C5,21
2400 DATA 1C,40,11,12,C2,06,04,F5,C5,D5,CD,0C,00,FB,D1,C1
2410 DATA 4F,1A,B9,20,08,F1,13,23,10,ED,C1,AF,C9,F1,C1,AF
2420 DATA 3C,C9,3A,1B,C2,FE,00,20,04,3E,FF,18,01,AF,32,1B
2430 DATA C2,C9,E5,DD,E1,DD,46,02,DD,7E,03,FE,02,28,0E,FE
2440 DATA 00,20,05,78,32,19,C2,C9,78,32,18,C2,C9,78,32,1A
2450 DATA C2,C9,4F,50,4C,4C,00,00,01,03,00,00,00,00,00,00
2460 DATA 00,FF,FF,FF,FF,FF,FF,FF,1B,19,FF,16,14,FF,22,0F
2470 DATA 26,24,27,25,FF,20,1E,07,FF,11,10,0E,0C,12,FF,0A
2480 DATA 1F,08,06,FF,17,03,04,18,13,23,21,09,0B,0D,FF,00
2490 DATA 1C,02,15,05,1D,1A,01,FF,FF,FF,FF,FF,FF,FF,FF,FF
2500 DATA FF,FF,FF,FF,FF,FF,FF,01,10,20,31,43,57,6B,81,98
2510 DATA B0,CA,E5,END
```

MSX2+

Appendix

1 8方向スクロールシューティングゲーム

MSX2+のハードウエアスクロール機能を使ったゲームを紹介します.

このゲームは自分の宇宙船を8方向に操作し，向かってくる敵キャラを撃ち落としながら，宇宙空間に浮かんでいる4つの敵母艦を倒すことを目的としたゲームです．途中3機編隊の敵戦闘機がおそってきますが，うまくそれを避けて目的を遂行してください.

操作法は，ジョイスティックまたは，キーボードのカーソルキーとスペースキーで遊ぶことができます．自機は3機です．敵機は弾を撃ってきたりはしませんが，編隊飛行で攻めてきたり，また自機よりもスピードが速く，執ように追跡してくるものもありますので，うまく避けながら，進んでいってください.

敵の母艦は，何発か弾を当てなければ倒すことができません．また，4つの母艦の位置を捜すためのMAP機能も付いています．ゲーム開始時には，画面に表示されています．赤いドットが，敵母艦の位置で，白いドットが，自分のいる位置です.

ゲーム中に自分が何処にいるのかわからなくなった時は，CTRL+STOPキー(CTRLキーを押しながらSTOPキーを押す)を押せば，ゲームを一時中断し，MAPを画面に表示することができます．何かほかのキーを押せば，ゲームに戻ることができます.

■ プログラムについて

このプログラムでは，スピードを実現するため，ほとんどの重要なルーチンはマシン語になっていますが，マシン語も含めたプログラムの構造は，だいたい次のようになっています.

- キー入力(ジョイスティック)センスルーチン
- 8方向スクロールルーチン
- 弾の発射と弾の移動ルーチン
- 敵キャラクタの移動ルーチン
- 弾，自機，敵キャラクタ，ボスキャラクタの当り判定ルーチン
- ボスキャラクタ表示ルーチン

このゲームのなかでは，8ドットごとの上下左右スクロールをしています．

MSX2で，スプライトを使ったアプリケーションで，スクロールの処理をする場合には，気を付けなければならないことがあります．X軸方向にスクロールさせた場合，スクロールと一緒にスプライトも画面上の表示位置がスクロールしていってしまいます．縦方向にスクロールさせた場合は，スプライトの表示位置をスクロールした分だけ元に戻す作業をしなければなりません．具体的には，スクロールさせる前にスプライトの表示を禁止し，スクロールを実行したのち，スプライトのY座標からスクロールして移動した分を差し引くということになります．このゲームでは，自機の画面上の位置は固定されていますから，いつも画面の中央にあるようにすればよいわけです．横方向へのスクロールではスプライトの表示位置は変化しません．

■ スクロールレジスタの設定方法

BASCIにはSET SCROLLステートメントが用意されていますから，たいていの場合はSET SCROLLだけで間に合うと思われますが．ゲーム等を作る場合は，処理が遅くてどうしようもない場合も考えられます．このプログラムではスクロールの処理を自前のルーチンで実現しています．スクロールレジスタに関しては前に紹介したとおりですが，ここでもう一度詳しく説明しましょう．

■ 水平スクロールレジスタ

水平スクロールレジスタは，右方向へ8ドットごとに移動させることのできるレジスタ(VDP(27))と，左方向へ1ドットごとに8ドット分移動させることのできるレジスタ(VDP(28))の2つに分かれています．

今回のゲームでは8ドットごとのスクロールですから，1つ目のレジスタ(VDP(27))のみにデータを設定していけばいいわけです．しかし8ドット以下の移動量でスクロールさせるときは，そうはいきません．

では，水平スクロール用の2つのレジスタに，どのように値を書き込めば，1ドットごとの水平方向スクロールができるようになるのか，簡単なBASICのサンプルプログラムで説明しましょう．

リスト1　左方向へのスクロール

```
100 J=6:I=I+1:IF I=32 THEN I=0
110 IF (VDP(-2) AND &B01000000)=0 THEN GOTO 110
120 VDP(27)=I:VDP(28)=7
130 FOR W=1 TO 180:NEXT W
140 IF (VDP(-2) AND &B01000000)=0 THEN GOTO 140
150 VDP(28)=J:J=J-1:IF J=-1 THEN J=6:GOTO 100
160 FOR W=1 TO 180:NEXT W
170 GOTO 140
```

リスト2　右方向へのスクロール

```
100 VDP(26)=VDP(26)OR &B00000010
110 J=1:I=I-1:IF I=-1 THEN I=31
120 IF (VDP(-2) AND &B01000000)=0 THEN GOTO 120
130 VDP(27)=I:VDP(28)=0
140 FOR W=0 TO 30:NEXT W
150 IF (VDP(-2) AND &B01000000)=0 THEN GOTO 150
160 VDP(28)=J:J=J+1:IF J=8 THEN J=1:GOTO 110
170 FOR W=0 TO 90:NEXT W
180 GOTO 150
```

まず左方向にスクロールする場合を考えて見ましょう．左方向にスクロールさせるわけですから，27番レジスタに値を書き込みます．左方向ですから，1を書き込めばよいわけです．そうすると，左に8ドット画面がスクロールします．しかし，これでは1ドットごとのスクロールにはなりません．そこで，28番レジスタに，7を書き込んで，右方向に7ドットスクロールさせます．これで，見かけ上1ドット左にスクロールしたことになります．

同様に2ドット左にスクロールしたい場合は，27番レジスタに1を，28番レジスタに6を書き込めばいいわけです．

サンプルプログラムでは28番レジスタへ書き込む値を順次小さくしながら8ドットスクロールさせ，また，27番レジスタに書き込むという処理を繰り返し行って，連続的にスクロールさせています．

右方向へのスクロールも左方向へのスクロールと同じように設定します．右方向の場合は，レジスタに与えるデータを少しずつ大きくしていくようになります．

■ 垂直帰線期間

110行は，VDPが垂直帰線期間であるかどうかを調べるためのものです．ためしにこの行を削除してみると，スクロールするごとに画面が乱れてしまうのがわかると思います．水平スクロールレジスタへのデータの書き込みをする際には，かならずこのフラグが1になっていることを確認してから書き込むようにしなければなりません．

垂直帰線期間というのは，CRTが画面を1画面分のデータを書き終えてから，その次の画面を描き始めるまで間の期間であることを示しています．

垂直帰線期間以外の期間，つまり，CRTが画面を描いている途中で，水平スクロールレジスタに値を書き込むと，画面が乱れてしまいます．これは，VDPが，画面を描いている間でも，水平スクロールレジスタにデータが書き込まれると，すぐに書き込まれた値に従った画像データを出力してしまい，画面の途中から，横方向の位置が一時的にずれてしまうために起こるものです．垂直帰線期間中に水平スクロールレジスタの値をを書き換えれば，乱れることはありません．

27番レジスタと28番レジスタへの書き込みをこの垂直帰線期間中に同時に行えば，いったん左側に8ドットスクロールして右側に7ドット戻すという処理も，画面上にはまったく現われなくなります．

サンプルプログラムでは，実際には少しずつ画面にノイズがかかっているようにちらついてしまいます．これば，120行で，2つのレジスタに値を書き込んでいますが，そのときのタイムラグで，2つめのVDPコマンドの実行が，垂直帰線期間を過ぎてしまってから行われていることから起きるものです．

マシン語で組まれたプログラムであれば，2つのレジスタに瞬時にデータをセットすることができるので，画面も乱れず，きちんと1ドットごとにスクロールしているように見えますが，同様のことをBASICで行うのは難しくなります．いちおうこのサンプルプログラムは，スクロールの動作原理を理解するためのものと思ってください．

右方向へのスクロールのサンプルプログラムVDP(26)(レジスタ#25)のビット1，ビット0は，水平スクロール時の効果を決定するビットです．SET SCROLLコマンドのMASKと2PAGEパラメータに当たります．

このサンプルプログラムでは，画面左端8ドットをボーダーカラーで隠すように設定しています．

■ 垂直スクロールレジスタ

垂直スクロールレジスタは，VDP(24)(レジスタ#23)に当たります．このレジスタに値を0～255までの値を書き込むことによって表示開始ラインを変更することができます．ただし，スクロールは，Y方向に256ライン単位で行われますから，スプライトアトリビュートテーブルや，カラーテーブルなどは，他のページに移動させるか，使わないようにする必要があります．

垂直スクロールでは，水平スクロールのように，垂直帰線期間にコマンドを書き込む必要はありません．

リスト3　下方向へのスクロール

```
100 VDP(24)=X
110 X=X+1:IF X=256 THEN X=0
120 FOR W=1 TO 100:NEXT W
130 GOTO 100
```

リスト4　上方向へのスクロール

```
100 VDP(24)=X
110 X=X-1:IF X=-1 THEN X=255
120 FOR W=1 TO 100:NEXT W
130 GOTO 100
```

リスト　8方向スクロール・シューティングゲーム

```
1000 CLEAR 100,&HCF7F:DEFINT A-Z:SCREEN 5,2,0:COLOR 15,0,0:CLS:HS=1000
1010 RESTORE 2810:FOR I=&HD240 TO&HDDA6:READ K$:K=VAL("&H"+K$):POKE I,
K:NEXT
1020 DEFUSR=&HD240:DEFUSR1=&HD243:DEFUSR2=&HD246:DEFUSR3=&HD249
1030 OPEN"grp:" FOR OUTPUT AS#1:STOP OFF:ON STOP GOSUB 1580
1040 SET PAGE0,2:CLS:RESTORE 2350
1050 FOR J=&H4800 TO&H5EFF STEP128:FOR I=J TOJ+23
1060 READ A$:A=VAL("&H"+A$):VPOKE I,A:NEXT I,J
1070 RESTORE 1710:FOR I=&H7800 TO&H7BFF:READ A:VPOKE I,A:NEXT
1080 RESTORE 1690:FOR I=0 TO511 STEP16:READ A:FOR J=0 TO15:VPOKE I+J,A
:NEXT J,I
1090 R=1:N=3:SC=0
1100 RESTORE 1670:FOR I=&HD200 TO&HD227:POKE I,0:NEXT
1110 FOR I=&HD200 TO&HD20F+((R-1) MOD7)*4 STEP4:READ X,Y
1120 POKE I+1,X:POKE I+2,Y:POKE I+3,20:NEXT
1130 SET PAGE0,3:CLS:X=0:Y=0:GOSUB 1660
1140 X=X+INT(RND(1)*2560):IF X>255 THEN Y=Y+X¥256:X=X MOD256:IF Y>211
THEN 1160
1150 PSET (X,Y),10:GOTO 1140
1160 COPY(0,0)-(255,211),3 TO(0,0),0:COPY(0,0)-(255,211),3 TO(0,0),1
1170 VDP(5)=231:VDP(6)=47:VDP(12)=2:VDP(9)=VDP(9) OR2
1180 RESTORE 1680:FOR I=&HD228 TO&HD232:READ A:POKE I,A:NEXT
1190 FOR I=&HCF80 TO&HD1FF:POKE I,0:NEXT
1200 SET PAGE1,1:PRESET(96,50):PRINT #1,"ROUND";R
1210 POKE &HD233,33-(R MOD7)*3:POKE &HCF80,1:POKE &HCF8A,0
1220 PRESET(104,80):PRINT #1,"REST";N
1230 SET PAGE1,1:PRESET(72,150):PRINT #1,"HIT SPACE KEY!":GOSUB 1580
1240 SET PAGE0,0:COLOR 15:VDP(9)=VDP(9) AND253:STOP ON:K=1:GOTO 1260
1250 K=STICK(0)+STICK(1):ON K+1 GOTO 1350,1260,1270,1280,1290,1300,131
0,1320,1330
1260 P=0:Q=-6:GOTO 1340
1270 P=4:Q=-4:GOTO 1340
1280 P=6:Q=0:GOTO 1340
1290 P=4:Q=4:GOTO 1340
1300 P=0:Q=6:GOTO 1340
1310 P=-4:Q=4:GOTO 1340
1320 P=-6:Q=0:GOTO 1340
1330 P=-4:Q=-4
1340 W=K*2-2
1350 POKE &HCF8A,W:POKE &HCF86,P AND255:POKE &HCF88,Q AND255
1360 IF STRIG(0)+STRIG(1)=0 THEN 1410
1370 IF PEEK(&HCFA0)=0 THEN POKE &HCFA0,2:GOTO 1410
```

```
1380 IF PEEK(&HCFC0)=0 THEN POKE &HCFC0,2:GOTO 1410
1390 IF PEEK(&HCFE0)=0 THEN POKE &HCFE0,2:GOTO 1410
1400 IF PEEK(&HD000)=0 THEN POKE &HD000,2
1410 A=USR1(0):A=USR3(0):A=USR(0):A=USR2(0):A=USR1(0)
1420 IF PEEK(&HD22F)=0 THEN 1520 ELSE IF PEEK(&HCF80)=129 THEN 1250
1430 FOR I=0 TO21:A=USR1(0):A=USR(0):A=USR2(0):A=USR1(0):FOR J=0 TO99:
NEXT J,I
1440 SC=SC+PEEK(&HD231)*1280+PEEK(&HD230)*5:IF SC>99999999# THEN SC=99
999999#
1450 STOP OFF:GOSUB 1660:N=N-1:IF N THEN 1160
1460 SET PAGE0,0:IF SC<=HS THEN 1480
1470 HS=SC:PRESET(52,160):PRINT #1,"YOUR SCORE IS BEST."
1480 PRESET(88,70):PRINT #1,"GAME  OVER"
1490 PRESET(48,120):PRINT #1,"HI-SCORE:";:PRINT #1,USING"########";HS
1500 PRESET(72,130):PRINT #1,"SCORE:";:PRINT #1,USING"########";SC
1510 IF STRIG(0)+STRIG(1) THEN 1090 ELSE 1510
1520 IF PEEK(&HD240) THEN 1250
1530 STOP OFF:GOSUB 1660:R=R+1:IF R>99 THEN R=1
1540 SC=SC+PEEK(&HD231)*1280+PEEK(&HD230)*5+R*1000
1550 IF SC>99999999# THEN SC=99999999#
1560 SET PAGE 0,0:PRESET(72,50):PRINT #1,USING"ROUND## CLEAR!";R
1570 PRESET(72,140):PRINT #1,"SCORE:";:PRINT #1,USING"########";SC:GOT
O 1130
1580 SET PAGE1,1:GOSUB 1660
1590 LINE(185,63)-(250,192),15,B:LINE(186,64)-(249,191),1,BF
1600 G=(PEEK(&HD229)*256+PEEK(&HD228)+128)/32:G=G MOD64
1610 H=(PEEK(&HD22B)*256+PEEK(&HD22A)+106)/32:H=H MOD128:PSET(186+G,64
+H),10
1620 FOR L=&HD200 TO&HD227 STEP4:IF PEEK(L+3)=0 THEN 1640
1630 G=PEEK(L+1)*8:H=PEEK(L+2)*8:PSET(186+G,64+H),8
1640 NEXT
1650 IF STRIG(0)+STRIG(1)=0 THEN 1650 ELSE RETURN
1660 VDP(24)=0:VDP(27)=0:VDP(28)=0:RETURN
1670 DATA 2,6,4,5,6,10,1,9,3,12,7,3,5,13,6,1,1,14,1,2
1680 DATA 0,4,0,8,0,8,0,1,0,0,9
1690 DATA 47,100,47,100,47,100,47,100,47,100,47,100,47,100,47,100
1700 DATA 43,43,40,41,36,37,35,35,35,35,35,35,35,35,38,40
1710 DATA 1,3,3,19,19,22,20,61,60,53,117,117,124,126,119,51
1720 DATA 128,192,192,200,200,104,40,60,60,44,46,46,62,126,238,204
1730 DATA 0,0,0,0,0,1,3,2,3,2,2,2,3,1,0,0
1740 DATA 0,0,0,0,0,128,192,192,192,192,192,192,192,128,0,0
1750 DATA 0,0,3,30,63,123,246,252,216,48,98,101,99,62,29,1
1760 DATA 64,158,62,126,254,252,57,50,180,124,216,184,248,240,224,192
1770 DATA 0,0,0,0,0,0,1,3,7,15,29,26,28,0,0,0
```

```
1780 DATA 0,0,0,0,0,0,192,192,64,128,0,0,0,0,0,0
1790 DATA 0,124,255,255,49,127,224,206,192,224,127,49,255,255,124,0
1800 DATA 0,0,128,248,128,224,62,159,31,62,224,128,248,128,0,0
1810 DATA 0,0,0,0,0,0,31,49,63,31,0,0,0,0,0,0
1820 DATA 0,0,0,0,0,0,192,96,224,192,0,0,0,0,0,0
1830 DATA 1,29,62,99,97,104,52,216,253,246,123,63,30,3,0,0
1840 DATA 192,224,240,248,184,216,124,52,58,57,252,254,126,62,158,64
1850 DATA 0,0,0,28,30,23,11,7,2,1,0,0,0,0,0,0
1860 DATA 0,0,0,0,0,0,128,192,192,192,0,0,0,0,0,0
1870 DATA 51,119,126,124,117,116,53,61,61,20,22,19,19,3,3,1
1880 DATA 204,238,126,62,46,46,44,60,60,40,104,200,200,192,192,128
1890 DATA 0,0,1,3,2,3,2,2,2,3,1,0,0,0,0,0
1900 DATA 0,0,128,192,192,192,192,192,192,192,128,0,0,0,0,0
1910 DATA 3,7,15,31,29,27,62,45,76,156,63,127,126,124,121,2
1920 DATA 128,184,124,198,166,70,12,27,63,111,222,252,120,192,0,0
1930 DATA 0,0,0,0,0,0,1,2,3,3,0,0,0,0,0,0
1940 DATA 0,0,0,56,88,184,240,224,192,128,0,0,0,0,0,0
1950 DATA 0,0,1,31,1,7,124,249,248,124,7,1,31,1,0,0
1960 DATA 0,62,255,255,140,254,7,115,3,7,254,140,255,255,62,0
1970 DATA 0,0,0,0,0,0,3,6,7,3,0,0,0,0,0,0
1980 DATA 0,0,0,0,0,0,248,140,252,248,0,0,0,0,0,0
1990 DATA 2,121,124,126,127,56,153,72,44,62,27,29,31,15,7,3
2000 DATA 0,0,192,120,252,222,111,63,91,44,12,140,248,112,128,128
2010 DATA 0,0,0,0,0,7,6,7,3,1,0,0,0,0,0,0
2020 DATA 0,0,0,0,0,0,128,192,160,208,240,112,0,0,0,0
2030 DATA 0,0,0,0,0,0,1,3,7,7,3,1,0,0,0,0
2040 DATA 0,0,0,0,0,0,128,192,224,224,192,128,0,0,0,0
2050 DATA 4,14,13,14,29,30,31,30,62,60,56,48,96,67,60,0
2060 DATA 0,0,0,128,64,160,80,168,84,10,1,14,112,128,0,0
2070 DATA 0,7,31,61,113,99,228,201,250,201,228,99,113,61,31,7
2080 DATA 0,192,240,120,28,140,78,38,190,38,78,140,28,120,240,192
2090 DATA 0,7,31,60,112,107,228,201,202,201,228,107,112,60,31,7
2100 DATA 0,192,240,120,28,172,78,38,166,38,78,172,28,120,240,192
2110 DATA 0,0,28,62,61,58,21,11,11,21,58,61,62,28,0,0
2120 DATA 0,0,56,124,188,92,168,208,208,168,92,188,124,56,0,0
2130 DATA 0,3,7,7,4,59,116,117,117,116,59,4,7,7,3,0
2140 DATA 0,192,224,224,32,222,47,175,175,47,222,32,224,224,192,0
2150 DATA 1,29,61,123,119,247,227,225,227,229,246,119,117,54,19,1
2160 DATA 128,184,188,222,238,239,199,135,199,167,111,238,174,108,200,
128
2170 DATA 7,31,63,124,121,249,241,247,237,220,183,51,62,61,3,7
2180 DATA 224,200,28,254,252,249,251,251,247,135,143,62,254,252,248,22
4
2190 DATA 7,31,63,124,0,62,109,219,219,109,62,0,124,63,31,7
```

```
2200 DATA 224,248,252,62,14,54,120,255,255,120,54,14,62,252,248,224
2210 DATA 7,3,61,62,51,183,220,237,247,241,249,121,124,63,15,3
2220 DATA 224,248,252,254,62,143,135,247,251,251,249,252,254,28,200,22
4
2230 DATA 1,19,54,117,119,246,229,227,225,227,247,119,123,61,29,1
2240 DATA 128,200,108,174,238,111,167,199,135,199,239,238,222,188,184,
128
2250 DATA 7,31,63,127,124,241,225,239,223,223,159,63,127,56,19,7
2260 DATA 224,192,188,124,204,237,59,183,239,143,159,158,62,252,248,22
4
2270 DATA 7,31,63,124,112,108,30,255,255,30,108,112,124,63,31,7
2280 DATA 224,248,252,62,0,124,182,219,219,182,124,0,62,252,248,224
2290 DATA 7,19,56,127,63,159,223,223,239,225,241,124,127,63,15,3
2300 DATA 224,248,252,62,158,159,143,239,183,59,237,204,124,188,192,22
4
2310 DATA 0,0,0,0,3,7,10,13,6,9,6,3,0,0,0,0
2320 DATA 0,0,0,0,128,96,160,208,176,176,96,128,0,0,0,0
2330 DATA 7,24,39,92,58,215,234,157,115,173,78,117,57,62,29,3
2340 DATA 192,240,44,214,218,109,93,171,183,213,210,238,92,184,240,192
2350 DATA 00,00,00,00,00,00,00,00,04,40,04,40,04,40,04,40,00,00,00,00,
00,00,00,00
2360 DATA 00,00,00,00,00,00,00,00,44,00,40,09,90,04,00,44,00,00,00,00,
00,00,00,00
2370 DATA 00,00,00,44,40,00,00,04,40,00,44,97,79,44,00,04,40,00,00,04,
44,00,00,00
2380 DATA 00,00,04,40,00,00,00,44,40,04,44,70,07,44,40,04,44,00,00,00,
04,40,00,00
2390 DATA 00,00,44,00,00,00,00,44,00,04,40,09,90,04,40,00,44,00,00,00,
00,44,00,00
2400 DATA 00,04,40,00,00,00,04,04,00,04,44,97,79,44,40,00,40,40,00,00,
00,04,40,00
2410 DATA 00,04,40,00,00,00,04,40,00,44,44,70,07,44,44,00,04,40,00,00,
00,04,40,00
2420 DATA 00,04,40,00,00,00,04,44,00,44,40,09,90,04,44,00,44,40,00,00,
00,04,40,00
2430 DATA 00,04,40,00,00,00,04,44,00,44,44,97,79,44,44,00,44,40,00,00,
00,04,40,00
2440 DATA 00,04,04,00,00,00,04,44,00,44,44,70,07,44,44,00,44,40,00,00,
00,40,40,00
2450 DATA 00,04,04,00,00,00,04,40,40,04,40,09,90,04,40,04,04,40,00,00,
00,40,40,00
2460 DATA 00,04,04,00,05,50,04,40,40,04,44,97,79,44,40,04,04,40,05,50,
00,40,40,00
```

```
2470 DATA 00,00,44,00,55,55,04,04,44,00,44,70,07,44,00,44,40,40,55,55,
00,44,00,00
2480 DATA 00,04,44,00,55,55,04,04,44,00,04,40,04,40,00,44,40,40,55,55,
00,44,40,00
2490 DATA 00,04,44,05,55,55,50,44,44,40,04,40,04,40,04,44,44,05,55,55,
50,44,40,00
2500 DATA 04,44,44,05,55,55,55,04,44,44,04,00,00,40,44,44,40,55,55,55,
50,44,44,40
2510 DATA 74,44,00,00,05,55,55,50,44,40,44,00,00,44,04,44,05,55,55,50,
00,00,44,47
2520 DATA 74,44,04,44,07,77,77,77,04,04,00,88,88,00,40,40,77,77,77,70,
44,40,44,47
2530 DATA 74,44,0B,B4,05,55,55,55,50,40,88,88,88,88,04,05,55,55,55,50,
4B,B0,44,47
2540 DATA 74,44,04,44,05,55,55,55,04,06,88,88,88,F8,80,40,55,55,55,50,
44,40,44,47
2550 DATA 74,44,0B,B4,05,55,55,55,04,06,88,88,88,88,80,40,55,55,55,50,
4B,B0,44,47
2560 DATA 74,44,04,44,05,55,55,50,40,66,88,88,88,8F,88,04,05,55,55,50,
44,40,44,47
2570 DATA 74,44,0B,B4,05,05,55,50,40,66,88,88,88,8F,88,04,05,55,50,50,
4B,B0,44,47
2580 DATA 44,44,04,44,05,55,55,50,40,66,68,88,88,8F,88,04,05,55,55,50,
44,40,44,44
2590 DATA 44,44,0B,B4,00,00,00,00,40,66,68,88,88,8F,88,04,00,00,00,00,
4B,B0,44,44
2600 DATA 44,44,04,44,04,04,07,77,04,06,66,88,88,88,80,40,77,70,40,40,
44,40,44,44
2610 DATA 44,44,00,00,04,04,04,44,04,06,66,66,88,88,80,40,44,40,40,40,
00,00,44,44
2620 DATA 44,40,44,44,44,04,07,74,40,40,66,66,66,66,04,04,47,70,40,44,
44,44,04,44
2630 DATA 44,04,44,44,44,04,04,44,44,04,00,66,66,00,40,44,44,40,40,44,
44,44,40,44
2640 DATA 40,44,44,44,44,04,04,44,44,00,44,00,00,44,00,44,44,40,40,44,
44,44,44,04
2650 DATA 04,44,44,44,44,04,04,44,00,44,00,00,00,00,44,00,44,40,40,44,
44,44,44,40
2660 DATA 04,44,44,44,44,04,04,40,44,49,00,00,00,00,94,44,04,40,40,44,
44,44,44,40
2670 DATA 00,44,44,44,44,04,04,40,04,47,99,99,99,99,74,40,04,40,40,44,
44,44,44,00
2680 DATA 05,04,44,44,44,00,04,40,40,44,77,77,77,77,44,04,04,40,00,44,
44,44,40,50
```

```
2690 DATA 04,50,44,44,44,04,04,00,44,04,00,00,00,00,40,44,00,40,40,44,
44,44,05,40
2700 DATA 04,45,00,44,44,04,00,50,44,49,00,00,00,00,94,44,05,00,40,44,
44,00,54,40
2710 DATA 00,44,55,00,00,00,55,50,04,47,99,99,99,99,74,40,05,55,00,00,
00,55,44,00
2720 DATA 00,44,45,55,55,55,55,40,40,44,77,77,77,77,44,04,04,55,55,55,
55,54,44,00
2730 DATA 00,04,44,45,55,55,44,00,44,04,00,00,00,00,40,44,00,44,55,55,
54,44,40,00
2740 DATA 00,00,00,44,44,44,40,00,44,49,00,00,00,00,94,44,00,04,44,44,
44,00,00,00
2750 DATA 00,00,00,04,44,44,40,00,44,47,99,99,99,99,74,44,00,04,44,44,
40,00,00,00
2760 DATA 00,00,00,00,44,44,00,00,44,40,77,77,77,77,04,44,00,00,44,44,
00,00,00,00
2770 DATA 00,00,00,00,00,00,00,00,04,40,00,00,00,00,04,40,00,00,00,00,
00,00,00,00
2780 DATA 00,00,00,00,00,00,00,00,04,44,00,00,00,00,44,40,00,00,00,00,
00,00,00,00
2790 DATA 00,00,00,00,00,00,00,00,00,44,40,00,00,04,44,00,00,00,00,00,
00,00,00,00
2800 DATA 00,00,00,00,00,00,00,00,00,00,44,40,04,44,00,00,00,00,00,00,
00,00,00,00
2810 DATA C3,7C,D4,C3,B7,D2,C3,A7,DA,ED,5F,5F,3A,36,D2,83,32,36,D2,21,
32,D2,35,C0
2820 DATA CD,A1,D2,79,B7,20,06,3E,01,32,32,D2,C9,3A,33,D2,32,32,D2,ED,
5F,E6,0E,16
2830 DATA 00,5F,21,91,D2,19,46,23,4E,C5,CD,A1,D2,38,10,C1,DD,71,00,DD,
70,1E,3A,36
2840 DATA D2,DD,77,09,10,EB,C9,C1,C9,01,07,01,07,01,09,01,07,03,07,03,
09,03,08,03
2850 DATA 0A,DD,21,20,D0,11,20,00,01,0B,0B,DD,7E,00,B7,C8,0D,DD,19,10,
F6,37,C9,DD
2860 DATA 21,20,D0,06,0B;DD,7E,00,FE,87,38,05,C5,CD,EA,D2,C1,11,20,00,
DD,19,10,ED
2870 DATA DD,7E,00,FE,8B,C0,CD,EA,D2,DD,7E,0D,87,87,21,FF,D1,16,00,5F,
19,DD,7E,1D
2880 DATA 77,C9,CD,A2,D3,3A,80,CF,FE,81,20,0A,FD,21,80,CF,CD,6F,D3,D2,
3F,D3,3A,A0
2890 DATA CF,FE,82,20,0A,FD,21,A0,CF,CD,6F,D3,D2,3F,D3,3A,C0,CF,FE,82,
20,0A,FD,21
2900 DATA C0,CF,CD,6F,D3,D2,3F,D3,3A,E0,CF,FE,82,20,0A,FD,21,E0,CF,CD,
6F,D3,D2,3F
```

```
2910 DATA D3,3A,00,D0,FE,82,C0,FD,21,00,D0,CD,6F,D3,D8,FD,35,1D,20,13,
FD,7E,00,E6
2920 DATA 3F,FD,77,1E,16,00,5F,21,70,D4,19,7E,FD,77,00,DD,35,1D,C0,DD,
7E,00,E6,3F
2930 DATA DD,77,1E,16,00,5F,21,70,D4,19,7E,DD,77,00,C9,16,00,FD,5E,0B,
21,ED,D3,19
2940 DATA FD,7E,04,5F,86,23,B8,30,06,7E,83,B8,30,05,C9,57,79,BA,D8,23,
FD,7E,02,5F
2950 DATA 86,23,D9,B8,30,07,D9,7E,83,D9,B8,D9,C9,57,79,BA,D9,C9,DD,5E,
0B,16,00,21
2960 DATA ED,D3,19,DD,7E,04,5F,86,FE,6A,38,05,FE,C0,38,33,AF,23,47,7B,
86,FE,6A,38
2970 DATA 06,FE,C0,30,26,3E,69,23,4F,DD,7E,02,5F,86,FE,80,38,05,FE,C0,
38,15,AF,23
2980 DATA 08,7B,86,FE,80,38,06,FE,C0,30,08,3E,7F,D9,4F,08,47,D9,C9,F1,
C9,01,06,01
2990 DATA 06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,
06,01,06,01
3000 DATA 06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,
06,01,06,01
3010 DATA 06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,01,06,00,07,00,
07,00,07,00
3020 DATA 07,00,07,00,07,00,07,00,07,00,07,00,07,00,07,00,07,00,07,00,
07,00,07,00
3030 DATA 07,00,07,00,07,00,07,00,07,00,07,00,07,00,07,00,07,00,07,00,
07,03,04,03
3040 DATA 04,01,06,01,06,06,10,06,11,03,04,83,84,85,86,05,05,05,05,06,
21,A7,DD,22
3050 DATA 34,D2,06,11,DD,21,80,CF,C5,DD,7E,00,E6,3F,28,11,87,26,00,6F,
11,AD,D4,19
3060 DATA 7E,23,66,6F,01,A1,D4,C5,E9,01,20,00,DD,09,C1,10,DF,2A,34,D2,
36,D8,C9,C5
3070 DATA D4,D3,D5,06,D6,4C,D6,5E,D6,71,D6,1C,D7,69,D7,3E,D7,E4,D7,5C,
D8,DD,CB,00
3080 DATA 7E,20,18,DD,36,03,00,DD,36,04,31,DD,36,01,00,DD,36,02,3C,DD,
36,1D,01,DD
3090 DATA CB,00,FE,3A,86,CF,16,00,5F,FE,80,38,01,15,2A,28,D2,19,7C,E6,
07,67,22,28
3100 DATA D2,3A,88,CF,16,00,5F,FE,80,38,01,15,2A,2A,D2,22,2C,D2,19,7C,
E6,0F,67,22
3110 DATA 2A,D2,7D,C6,62,FE,D9,28,01,3D,32,A7,DD,32,AB,DD,3E,78,32,A8,
DD,32,AC,DD
3120 DATA 3A,8A,CF,87,87,32,8B,CF,32,A9,DD,C6,04,32,AD,DD,21,AF,DD,22,
34,D2,3A,80
```

```
3130 DATA D1,FE,8B,C8,01,00,0A,FD,21,24,D2,FD,7E,03,B7,28,4A,0C,FD,66,
01,2E,00,ED
3140 DATA 5B,28,D2,B7,ED,52,38,0C,7C,B7,20,37,CB,3D,7D,32,82,D1,18,0E,
7D,11,D2,FF
3150 DATA B7,ED,52,38,26,CB,2F,32,82,D1,FD,66,02,2E,00,ED,5B,2A,D2,B7,
ED,52,7D,38
3160 DATA 0A,11,D4,00,B7,ED,52,38,16,18,08,11,D2,FF,B7,ED,52,30,10,11,
FC,FF,FD,19
3170 DATA 10,A9,79,32,2F,D2,C9,CB,3F,18,02,CB,2F,32,84,D1,78,32,8D,D1,
AF,32,81,D1
3180 DATA 32,83,D1,3E,0B,32,80,D1,FD,7E,03,32,9D,D1,FD,7E,01,32,86,D1,
FD,7E,02,32
3190 DATA 88,D1,C9,DD,CB,00,7E,20,2A,DD,36,0B,40,DD,36,09,C0,DD,36,1D,
01,DD,36,03
3200 DATA 00,DD,36,01,00,DD,36,04,31,DD,36,02,3C,DD,36,0C,16,3A,8A,CF,
CD,5E,D9,DD
3210 DATA CB,00,FE,C3,D0,D9,DD,CB,00,7E,20,2F,DD,36,0D,12,21,47,D6,DD,
74,0F,DD,75
3220 DATA 0E,DD,74,11,DD,75,10,DD,36,12,01,DD,36,06,00,DD,36,05,00,DD,
36,08,00,DD
3230 DATA 36,07,00,DD,36,09,C8,DD,CB,00,FE,CD,D0,D9,DD,35,0D,C0,DD,36,
00,00,C9,02
3240 DATA 7C,01,78,00,DD,CB,00,7E,20,E9,DD,36,0D,02,21,5B,D6,18,B8,02,
78,00,DD,CB
3250 DATA 00,7E,20,D7,2A,30,D2,23,22,30,D2,DD,36,0D,03,18,9F,DD,CB,00,
7E,20,51,DD
3260 DATA CB,00,FE,DD,36,0D,10,06,08,21,EC,D6,FD,21,A0,D1,FD,36,00,80,
FD,36,09,C0
3270 DATA FD,36,0B,7C,FD,36,01,00,FD,36,03,00,DD,7E,02,86,23,FD,77,02,
DD,7E,04,86
3280 DATA 23,FD,77,04,5E,23,56,23,FD,72,06,FD,73,05,5E,23,56,23,FD,72,
08,FD,73,07
3290 DATA 11,0C,00,FD,19,10,C1,C9,DD,35,0D,20,05,DD,36,00,00,C9,DD,E5,
DD,21,A0,D1
3300 DATA 06,08,C5,DD,7E,00,C4,D0,D9,C1,11,0C,00,DD,19,10,F1,DD,E1,C9,
08,02,00,00
3310 DATA 00,FF,0C,04,B4,00,4C,FF,0E,08,00,01,00,00,0C,0C,B4,00,B4,00,
08,0E,00,00
3320 DATA 00,01,04,0C,4C,FF,B4,00,02,08,00,FF,00,00,04,04,4C,FF,4C,FF,
DD,CB,00,7E
3330 DATA 20,19,CD,FA,D8,DD,36,0C,04,DD,36,0B,44,DD,36,09,C0,DD,36,1D,
01,DD,7E,0A
3340 DATA CD,5E,D9,C3,D0,D9,DD,CB,00,7E,20,F7,CD,FA,D8,DD,36,0C,08,21,
64,D7,DD,74
```

```
3350 DATA 0F,DD,75,0E,DD,74,11,DD,75,10,DD,36,12,01,DD,36,09,C8,18,CD,
01,50,01,54
3360 DATA 00,DD,CB,00,7E,20,2B,CD,FA,D8,DD,36,1D,01,DD,36,0C,05,21,DF,
D7,DD,74,0F
3370 DATA DD,75,0E,DD,74,11,DD,75,10,DD,36,12,01,DD,36,09,C8,DD,36,0D,
08,DD,7E,0A
3380 DATA 18,3F,DD,35,0D,20,3D,DD,36,0D,0C,01,0C,00,DD,7E,02,D6,3C,30,
04,ED,44,0E
3390 DATA 04,5F,DD,7E,04,D6,31,30,04,ED,44,06,08,FE,05,38,14,57,7B,FE,
05,38,0B,87
3400 DATA BA,30,0A,CB,3F,CB,22,BA,30,0A,78,18,01,79,DD,77,0A,CD,5E,D9,
C3,D0,D9,01
3410 DATA 48,01,4C,00,DD,CB,00,7E,20,18,CD,FA,D8,DD,36,09,C0,DD,36,1D,
01,DD,36,0C
3420 DATA 08,DD,36,0D,0E,DD,7E,0A,18,42,DD,35,0D,20,49,DD,36,0D,12,DD,
7E,02,D6,3C
3430 DATA 38,0E,FE,04,38,05,21,5A,D8,18,11,21,57,D8,18,0C,FE,FC,30,05,
21,54,D8,18
3440 DATA 03,21,57,D8,DD,7E,04,D6,31,38,08,FE,04,38,09,23,C3,40,D8,FE,
FC,30,01,2B
3450 DATA 7E,DD,77,0A,87,C6,58,DD,77,0B,DD,7E,0A,CD,5E,D9,C3,D0,D9,06,
04,02,08,00
3460 DATA 00,0A,0C,0E,DD,CB,00,7E,20,09,DD,36,0B,80,DD,CB,00,FE,C9,DD,
66,06,2E,00
3470 DATA ED,5B,28,D2,B7,ED,52,38,0B,7C,B7,20,40,CB,3D,DD,75,02,18,0E,
4D,11,D2,FF
3480 DATA B7,ED,52,38,30,CB,29,DD,71,02,DD,66,08,2E,00,ED,5B,2A,D2,B7,
ED,52,4D,38
3490 DATA 0E,11,D4,00,B7,ED,52,30,14,CB,39,DD,71,04,C9,11,D2,FF,B7,ED,
52,38,06,CB
3500 DATA 29,DD,71,04,C9,DD,36,00,00,C9,2C,F9,7F,21,2C,69,F9,21,1C,F1,
87,11,3C,71
3510 DATA F1,31,3C,F1,87,31,1C,71,F1,11,06,00,00,02,00,00,02,02,00,02,
02,04,02,02
3520 DATA 04,04,02,04,04,06,04,04,06,06,04,06,06,00,06,06,00,00,DD,CB,
00,FE,DD,36
3530 DATA 03,00,DD,36,01,00,DD,7E,09,E6,03,5F,3A,8A,CF,87,83,16,00,5F,
21,DA,D8,19
3540 DATA 7E,5F,87,C6,08,E6,0E,DD,77,0A,21,BA,D8,DD,7E,1E,87,87,87,83,
5F,19,DD,7E
3550 DATA 0A,B7,28,17,FE,08,28,13,7E,23,DD,77,02,DD,7E,09,CB,3F,CB,3F,
CB,3F,86,DD
3560 DATA 77,04,C9,DD,7E,09,CB,3F,CB,3F,CB,3F,86,23,DD,77,02,7E,DD,77,
04,C9,87,87
```

```
3570 DATA 16,00,5F,21,90,D9,19,5E,23,56,23,7E,23,66,6F,E5,DD,46,0C,21,
00,00,19,10
3580 DATA FD,DD,74,08,DD,75,07,D1,DD,46,0C,21,00,00,19,10,FD,DD,74,06,
DD,75,05,C9
3590 DATA 73,FF,00,00,7E,FF,35,00,9C,FF,64,00,CB,FF,82,00,00,00,8D,00,
35,00,82,00
3600 DATA 64,00,64,00,82,00,35,00,8D,00,00,00,82,00,CB,FF,64,00,9C,FF,
35,00,7E,FF
3610 DATA 00,00,73,FF,CB,FF,7E,FF,9C,FF,9C,FF,7E,FF,CB,FF,DD,CB,09,86,
DD,CB,09,7E
3620 DATA C4,19,DA,DD,CB,09,76,C4,4B,DA,DD,CB,09,5E,C4,82,DA,DD,CB,09,
46,C0,2A,34
3630 DATA D2,DD,7E,03,17,DD,7E,04,17,57,3A,2A,D2,82,FE,D9,28,01,3D,77,
23,DD,7E,01
3640 DATA 17,DD,7E,02,17,77,23,DD,7E,0B,77,23,23,22,34,D2,C9,DD,46,08,
DD,4E,07,DD
3650 DATA 66,04,DD,6E,03,09,0E,00,3A,88,CF,47,CB,28,CB,19,B7,ED,42,DD,
74,04,DD,75
3660 DATA 03,7C,FE,6A,D8,FE,E8,30,3C,FE,83,38,38,DD,36,00,00,F1,C9,DD,
46,06,DD,4E
3670 DATA 05,DD,66,02,DD,6E,01,09,0E,00,3A,86,CF,47,CB,28,CB,19,B7,ED,
42,DD,74,02
3680 DATA DD,75,01,7C,FE,80,D8,FE,E8,30,0A,FE,99,38,06,DD,36,00,00,F1,
C9,DD,CB,09
3690 DATA C6,C9,DD,35,12,C0,DD,66,11,DD,6E,10,7E,B7,20,07,DD,66,0F,DD,
6E,0E,7E,23
3700 DATA DD,77,12,7E,23,DD,77,0B,DD,74,11,DD,75,10,C9,21,03,03,22,06,
DD,CD,C7,DB
3710 DATA 3A,2E,D2,32,07,DD,26,00,01,00,00,11,00,01,CD,08,DD,3A,80,D1,
FE,8B,20,5F
3720 DATA 3A,82,D1,FE,E9,38,0A,ED,44,87,4F,3E,30,91,5F,18,14,FE,80,30,
4A,FE,69,38
3730 DATA 08,0E,00,87,ED,44,5F,18,04,0E,00,1E,30,3A,84,D1,FE,E9,38,0A,
ED,44,87,47
3740 DATA 3E,30,90,57,18,16,FE,6A,30,25,FE,52,38,0A,06,00,87,57,3E,D4,
92,57,18,04
3750 DATA 06,00,16,30,69,78,C6,90,67,3E,02,32,43,DD,3A,2E,D2,32,44,DD,
CD,45,DD,CD
3760 DATA 9A,DC,3A,28,D2,ED,44,CB,2F,CB,2F,CB,2F,ED,44,E6,1F,47,0E,1A,
F3,CD,47,00
3770 DATA FB,3A,28,D2,ED,44,E6,07,47,0E,1B,F3,CD,47,00,FB,3A,2A,D2,47,
0E,17,F3,CD
3780 DATA 47,00,FB,CD,93,DB,3A,2E,D2,B7,28,0F,AF,32,2E,D2,06,E7,0E,05,
F3,CD,47,00
```

```
3790 DATA FB,18,0D,3C,32,2E,D2,06,EF,0E,05,F3,CD,47,00,FB,06,02,0E,0B,
F3,CD,47,00
3800 DATA FB,06,2F,0E,06,F3,CD,47,00,FB,C9,3A,2E,D2,B7,20,17,01,02,1F,
F3,CD,47,00
3810 DATA 01,0B,00,F3,CD,47,00,01,06,0F,F3,CD,47,00,FB,C9,01,02,3F,F3,
CD,47,00,01
3820 DATA 0B,01,F3,CD,47,00,01,06,1F,F3,CD,47,00,FB,C9,21,03,03,22,06,
DD,3A,88,CF
3830 DATA B7,C8,FE,80,DA,38,DC,ED,44,16,00,5F,3A,2C,D2,67,3A,2A,D2,47,
0E,00,7C,B7
3840 DATA 28,0A,BB,38,0F,C6,D4,28,03,BB,38,24,78,C6,D4,67,CD,08,DD,C9,
7B,94,5F,78
3850 DATA C6,D4,67,83,6F,E5,CD,08,DD,E1,65,3A,2C,D2,16,00,5F,01,00,00,
CD,08,DD,C9
3860 DATA 78,C6,D4,ED,44,5F,80,6F,78,C6,D4,67,E5,CD,08,DD,E1,45,3A,2C,
D2,C6,D4,16
3870 DATA 00,5F,4A,62,CD,08,DD,C9,16,00,5F,3A,2C,D2,67,3A,2A,D2,47,0E,
00,78,B7,28
3880 DATA 0A,BB,38,0F,C6,D4,28,03,BB,38,24,7C,C6,D4,47,CD,08,DD,C9,7C,
ED,44,5F,7C
3890 DATA C6,D4,47,83,6F,E5,CD,08,DD,E1,45,3A,2A,D2,16,00,5F,4A,62,CD,
08,DD,C9,7C
3900 DATA C6,D4,ED,44,16,00,5F,84,6F,7C,C6,D4,47,E5,CD,08,DD,E1,65,3A,
2A,D2,C6,D4
3910 DATA 16,00,5F,01,00,00,CD,08,DD,C9,3E,02,32,F6,FA,21,02,02,22,43,
DD,3A,2E,D2
3920 DATA B7,28,13,21,A7,DD,11,00,72,01,80,00,F3,CD,5C,00,FB,01,00,E0,
18,11,21,A7
3930 DATA DD,11,00,76,01,80,00,F3,CD,5C,00,FB,01,00,E8,1E,20,DD,21,A7,
DD,C5,D5,DD
3940 DATA 7E,00,FE,D8,28,25,DD,7E,02,26,00,6F,29,29,29,11,20,01,CD,45,
DD,11,04,00
3950 DATA DD,19,D1,E1,01,20,00,09,44,4D,1D,C2,D5,DC,AF,32,F6,FA,C9,D1,
E1,C9,00,00
3960 DATA F3,C5,CD,85,DD,3A,07,00,4F,0C,3E,22,ED,79,3E,91,ED,79,0C,0C,
ED,61,3A,06
3970 DATA DD,ED,79,E1,AF,ED,69,ED,79,ED,61,3A,07,DD,ED,79,AF,ED,79,ED,
79,ED,59,ED
3980 DATA 51,ED,79,ED,79,3E,E0,ED,79,FB,C9,00,00,F3,C5,CD,85,DD,3A,07,
00,4F,0C,3E
3990 DATA 20,ED,79,3E,91,ED,79,0C,0C,97,ED,69,ED,79,ED,61,3A,43,DD,ED,
79,E1,AF,ED
4000 DATA 69,ED,79,ED,61,3A,44,DD,ED,79,AF,ED,59,ED,79,ED,51,ED,79,ED,
79,ED,79,3E
```

```
4010 DATA D0,ED,79,FB,C9,3E,02,CD,92,DD,E6,01,20,F7,AF,C3,92,DD,C5,ED,
4B,07,00,0C
4020 DATA ED,79,3E,8F,ED,79,3A,06,00,4F,0C,ED,78,C1,C9,B4,E1,DD,21,38,
00,FD,2A,C0
```

MSX 漢字ドライバ拡張BIOS MSX-MUSIC FM BIOS仕様

最後にマシン語を使って，プログラムを組まれる方のために，MSX 漢字ドライバの拡張 BIOS と，MSX-MUSIC の FM BIOS を掲載しておきます．

■ 漢字ドライバ拡張 BIOS

漢字ドライバの拡張 BIOS コールを行う場合には，デバイス番号（D レジスタに渡す値）を 11H として行います．

漢字ドライバはエントリアドレステーブルは持っていませんが，ファンクション番号を指定し，FFCAH 番地の呼び出すことにより，現在の画面モードを返したり，希望する画面モードに設定したりできます．

漢字の表示は BDOS コールや，BIOS の CHPUT をコールすることにより行うことができるので，いままでと変わりありません．［ ］内はレジスタを表わします（例：［A］は A レジスタ）．

モードの取得

現在の画面モードを数値で返す

ファンクション番号：［E］=0（モードの取得）
エントリパラメータ：［A」=インストール検出（下記参照）
リターンパラメータ：［A］=モード
=0　(ANK)
=1　(KANJI0)
=2　(KANJI1)
=3　(KANJI2)
=4　(KANJI3)

漢字ドライバがインストールされていない場合は[A]が不変で返ってきます．これを利用して，[A]を0FFHにして呼ぶことにより，漢字ドライバがそもそもインストールされているのか否かがわかります．

単にモードが知りたいだけなら，[A]を0にして呼ぶことをおすすめします．

モードの設定

画面のモードを設定する

ファンクション番号：[E]＝1(モードの設定)
エントリパラメータ：[A]＝モード
＝0　(ANK)
＝1　(KANJI0)
＝2　(KANJI1)
＝3　(KANJI2)
＝4　(KANJI3)
リターンパラメータ：なし

スクリーンを指定されたモードに設定します．漢字ドライバがインストールされていないと無視されます．

■ FM BIOS

このFM BIOSは，他のユーティリティソフトウェアから使用される際に必要となるエントリアドレスをまとめたものです．以下のエントリのためのラベル名とアドレス，その入出力の設定レジスタ，内容が変化するレジスタ，及びその機能について解説します．

WRTOPL

OPLLレジスタへのデータの書き込み

エントリアドレス　：4110H
エントリパラメータ：［A］＝OPLLMのレジスタ番号
　　　　　　　　　　［E］＝書き込むデータ
リターンパラメータ：なし
使用レジスタ　　　：全保存

Aレジスタで指定した番号のOPLLレジスタに，Eレジスタの内容を書き込みます．このエントリルーチン内では，割り込み禁止･解除をしていませんので，必要に応じてDI，EIを行う必要があります．

INIOPL

OPLLミュージックドライバーの環境を整備する

エントリアドレス　：4113H
エントリパラメータ：［HL］＝仕様ワークエリアの先頭アドレス(偶数)
リターンパラメータ：なし
使用レジスタ　　　：AF　BC　DE　HL　IX　IY

HLレジスタで指定されたアドレス(偶数)に，ミュージックドライバーで使用するワークエリアを設定し，すべてのミュージックワークエリア及びOPLLレジスタを初期化します．ワークエリアの先頭アドレス(偶数)は，SLTWRK (0FD09H-)に格納されます．HLレジスタで指定されたエリアがRAMでない場合の動作保証はありません．

すべてのエントリルーチンは，はじめにかならず1度はこの INIOPL を呼び出しておく必要があります．このルーチンを1度も呼び出さずに，他のエントリルーチンを呼び出した時は，その動作保証はありません．

EI 状態で RET します．

MSTART

音楽の演奏を開始する

エントリアドレス　：4116H
エントリパラメータ：［HL］＝ミュージックデータの先頭アドレス
　［A］＝エンドレスフラグ
　　＝0　無限ループ
　　＝1－254　繰り返し演奏回数の指定
　　＝255　動作保証なし
(HL)＝0EH　リズムモード　(FM6 音＋リズム部)
　12H　メロディモード(FM9 音)
※(HL)はデータとして持っていますから，新たに書き込む必要はありません．
リターンパラメータ：なし
使用レジスタ　：AF　BC　DE　HL　IX　IY

HL レジスタで指定されたアドレスに配置されているミュージックデータのヘッダをもとに，OPLL ドライバのワークエリアを音楽演奏用にセットします．H. TIMI フック(0FD9FH)から OPLDRV を呼び出すように，あらかじめセットしておかないと，音楽の演奏はしません．

EI 状態で RET します．

MSTOP

音楽演奏を中止する

エントリアドレス　：4119H
エントリパラメータ：なし
リターンパラメータ：なし
使用レジスタ　　　：AF　BC　DE　HL　IX　IY

現在出力しているOPLLのすべての音の発生を止め，OPLLドライバーのワークエリアも初期化します．
EI状態でRETします．

RDDATA

ROM内部音色データの読み出し

エントリアドレス　：411CH
エントリパラメータ：［HL］＝データ読み出し用ワークの先頭アドレス
　　　　　　　　　　［A］　＝音色ナンバー(0〜63)
リターンパラメータ：なし
使用レジスタ　　　：F(フラグ)

ROMに内蔵されている音色を読み出し，指定のワークエリアに格納します．

OPLDRV

OPLL ドライバーのエントリアドレス

エントリアドレス　：411FH
エントリパラメータ：なし
リターンパラメータ：なし
使用レジスタ　　：全保存

音楽演奏を実際に行う OPLL ドライバーのエントリアドレスです．H. TIMI フックを書き変えて，この OPLDRV を呼び出すようにすれば接続 OK です．ただし，かならず 1 度は INIOPL を呼び，すべてのワーク及び OPLL レジスタを初期化してから，OPLDRV を接続するようにしてください．この手順を踏まえて接続しない場合は，その後の動作が保証できません．

TSTBGM

演奏終了のチェック

エントリアドレス　：4122h
エントリパラメータ：なし
リターンパラメータ：［A］＝演奏状態
　　　　＝0　　演奏終了
　　　　<>0　　演奏中
使用レジスタ　　：AF

MSTART で演奏を開始した音楽が，現在演奏を継続中かを確認します．割り込み禁止・解除の設定はしていません．

———レイトレーシング・サンプルデータ———

巻頭の口絵で紹介しているレイトレーシングのサンプルデータを掲載しておきます．リスト2.7のレイトレーシング・プログラムの5000行以下のデータ文を以下のように書き換えて実行してください．

```
5000 ' data
5010 DATA   20,  40,  20
5020 DATA    0,   0,   0
5030 DATA   -8,   9,  -3
5040 DATA    6
5050 DATA    6
5060 DATA    2,   0,   2
5070 DATA   .2,  .2,  .2
5080 DATA    1,   2
5090 DATA   -2,   2,   2
5100 DATA   .2,  .2,  .2
5110 DATA    1,   3
5120 DATA   -6,   4,   2
5130 DATA   .2,  .2,  .2
5140 DATA    1,   4
5150 DATA   -2,   2,  -2
5160 DATA   .2,  .2,  .2
5170 DATA    1,   5
5180 DATA   -6,   4,  -6
5190 DATA   .2,  .2,  .2
5200 DATA    1,   6
5210 DATA    0,  -2,   0
5220 DATA   20,   1,  20
5230 DATA    0,  -1
5240 DATA    7
5250 DATA   .9,  .9,  .9
5260 DATA   .5,  .4,  .6
5270 DATA   .7,   6
5280 DATA   .0,  .9,  .0
5290 DATA   .5,  .4,  .6
5300 DATA   .7,   6
5310 DATA   .9,  .0,  .0
5320 DATA   .3,  .6,   0
5330 DATA    0,   0
5340 DATA   .9,  .9,  .9
5350 DATA   .3,  .6,   0
5360 DATA   .6,   8
5370 DATA   .0,  .0,  .9
5380 DATA   .3,  .6,   0
5390 DATA   .6,   8
5400 DATA   .9,  .9,  .9
5410 DATA   .3,  .6,   0
5420 DATA   .6,   6
5430 DATA   .0,  .0,  .0
5440 DATA   ,3,  .6,   1
5450 DATA   .9,   8
```

■ 参考文献

・誰にでもわかるデジタルサウンド入門

東亜音楽社発行・音楽之友社発売

・MSX2 テクニカルハンドブック　アスキー

・YM2413 アプリケーションマニュアル　ヤマハ

・V9938　MSX-VIDEO テクニカルデータブック　アスキー

・実習グラフィックス
・応用グラフィックス　アスキー

・NHK カラーテレビ教科書(上)　日本放送出版協会

■ プログラム協力

蔭山　哲也
須田　淳
染谷　哲人

MSX2＋ パワフル活用法

1989年1月21日　初版発行
1991年10月1日　第1版第8刷発行
定価1,240円(本体1,204円)

著　者　杉谷成一(すぎたにせいいち)
発行者　藤井　章生
発行所　株式会社アスキー
　〒107-24　東京都港区南青山6-11-1スリーエフ南青山ビル
振　替　東京4-161144
TEL　(03)3486-7111（大代表）
情報TEL　(03)3498-0299（ダイヤルイン）
出版営業部TEL　(03)3486-1977（ダイヤルイン）

制　作　株式会社GARO
印　刷　株式会社加藤文明社印刷所

編　集　竹内充彦

ISBN4-87148-345-2

●11348／11430